

Laporan Milestone III TBFO

Semantic Analysis

ParsingIsAllYouNeed



Muhammad Raihan Nazhim Oktana

13523021

13523021@std.stei.itb.ac.id

Shanice Feodora Tjahjono

13523097

13523097@std.stei.itb.ac.id

Faqih Muhammad Syuhada

13523057

13523057@std.stei.itb.ac.id

Muhammad Fathur Rizky

13523105

13523105@std.stei.itb.ac.id

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2025

Daftar Isi

BAB I: Deskripsi Tugas	2
BAB II: Landasan Teori	3
2.1 Analisis Sintaksis (Parsing)	3
2.2 Algoritma Recursive Descent	3
2.3 Parse Tree	3
2.4 Keterhubungan antara Parser dan Lexer	4
BAB III: Perancangan & Implementasi	5
3.1 Perancangan	5
3.1.1 Arsitektur Sistem	5
3.1.2 Representasi Data dan Struktur AST	5
3.2 Implementasi	6
3.2.1 Penyesuaian Lexer dengan Post-processing	6
3.2.2 Base AST Class	7
3.2.2 Program	8
3.2.3 Expression	9
3.2.4 Declaration	10
3.2.5 Statement	12
3.2.6 Parser	13
3.2.7 Tree Formatter	16
BAB IV: Pengujian	19
4.1 Pengujian Test Cases	19
BAB V: Kesimpulan dan Saran	76
5.1 Kesimpulan	76
5.2 Saran	77
Lampiran	77
Daftar Pustaka	77

BAB I: Deskripsi Tugas

Pada era modern, bahasa pemrograman memegang peran fundamental dalam pengembangan perangkat lunak serta sistem komputasi. Salah satu komponen penting dalam ekosistem pemrograman adalah compiler, yaitu perangkat lunak yang menerjemahkan kode sumber (*source code*) ke dalam representasi yang dapat dipahami dan dieksekusi oleh mesin. Proses kompilasi terdiri atas beberapa tahap berurutan yang masing-masing memiliki fungsi spesifik dalam mentransformasikan program. Setelah tahap analisis leksikal (lexical analysis) berhasil dilaksanakan pada Milestone I dan tahap analisis sintaksis (syntax analysis) berhasil dilaksanakan pada Milestone II, Tugas Besar Mata Kuliah Teori Bahasa Formal dan Otomata (IF2224-24) berlanjut ke Milestone III, yaitu tahap *semantic analysis*.

Bahasa Pascal-S merupakan subset dari bahasa pemrograman Pascal yang telah disederhanakan dengan tujuan memfasilitasi pembelajaran konsep *compiler construction* secara bertahap dan terstruktur. Dalam tugas besar ini, mahasiswa bertindak sebagai pengembang Pascal-S Compiler dan diharapkan dapat mengimplementasikan komponen-komponen inti *compiler front-end*.

Pada Milestone II, mahasiswa berfokus pada pembangunan parser yang merupakan bagian penting dari tahap *syntax analysis*. Parser bertugas memeriksa rangkaian token yang dihasilkan oleh lexer, untuk memastikan bahwa token-token tersebut tersusun sesuai dengan aturan tata bahasa formal (grammar) Pascal-S. Lebih jauh, parser dibangun menggunakan pendekatan *Recursive Descent Parsing*, yaitu metode *top-down parsing* yang mengimplementasikan setiap aturan produksi grammar sebagai prosedur rekursif.

Pada Milestone III, mahasiswa perlu mengimplementasi *semantic analyzer* yang bekerja dengan menelusuri parse tree menggunakan pola desain *Visitor* dan aturan *Attributed Grammar*. Mahasiswa harus membangun dan mengelola struktur data *Symbol Table* untuk menyimpan informasi mendetail mengenai identifier, termasuk tipe data dan lingkup (scope) keberadaannya. Proses ini bertujuan untuk melakukan pemeriksaan tipe (type checking) dan pemeriksaan lingkup (scope checking) guna memastikan konsistensi program, seperti mencegah operasi matematika pada tipe data yang tidak kompatibel atau penggunaan variabel yang belum dideklarasikan. Luaran akhir dari tahap ini adalah *Decorated Abstract Syntax Tree* (AST) yang telah dianotasi dengan informasi semantik yang lengkap sebagai persiapan menuju tahap kompilasi selanjutnya.

BAB II: Landasan Teori

2.1 Konsep Dasar Compiler dan Peran Semantic Analysis

Compiler merupakan perangkat lunak yang menerjemahkan program sumber berbahasa tingkat tinggi menjadi representasi yang dapat dieksekusi oleh mesin. Secara umum, proses kompilasi terdiri atas serangkaian tahapan berurutan yang meliputi analisis leksikal, analisis sintaktis, analisis semantik, pembentukan representasi menengah, optimisasi, dan akhirnya generasi kode. Pada tahap awal, analisis sintaktis memastikan bahwa program mengikuti bentuk gramatikal bahasa pemrograman; namun kepatuhan terhadap sintaks tidak menjamin bahwa sebuah program bermakna secara benar. Di sinilah peran semantic analysis menjadi krusial.

Semantic analysis bertugas memverifikasi bahwa setiap konstruksi program sesuai dengan aturan makna bahasa—misalnya memastikan bahwa variabel telah dideklarasikan sebelum digunakan, tipe data antara operasi dan operand konsisten, dan prosedur atau fungsi dipanggil dengan jumlah serta tipe argumen yang benar. Aho, Lam, Sethi, dan Ullman dalam *Compilers: Principles, Techniques, and Tools* (2007) menyatakan bahwa semantic analysis adalah tahap yang memastikan bahwa program “makes sense” dan dapat diterjemahkan dengan aman ke tahap berikutnya dalam kompilasi.

2.2 Attributed Grammar sebagai Dasar Analisis Semantik

Attributed Grammar, yang diperkenalkan oleh Donald Knuth pada tahun 1968, merupakan perluasan dari context-free grammar dengan menambahkan atribut pada simbol-simbol grammar. Atribut ini digunakan untuk mengekspresikan makna dari suatu konstruksi sintaksis serta memungkinkan proses kompilasi menghitung informasi semantik secara sistematis. Setiap produksi dalam grammar dilengkapi dengan aturan semantik yang menentukan bagaimana atribut sesuatu node diperoleh—baik dari child node (synthesized attributes) maupun dari parent node (inherited attributes).

Dalam implementasi compiler modern, terutama untuk bahasa dengan struktur blok seperti Pascal, attributed grammar memungkinkan penghitungan informasi tipe, hubungan antar-blok, maupun informasi referensial. Attributed grammar yang bersifat L-attributed banyak digunakan karena kompatibel dengan traversal pohon secara top-down dan bottom-up menggunakan metode visitor, sehingga ideal untuk implementasi berbasis AST.

2.3 Symbol Table sebagai Penyimpan Informasi Makna

Symbol table adalah struktur data inti dalam semantic analysis yang digunakan untuk menyimpan dan mengelola informasi mengenai identifier, termasuk nama variabel, tipe data, lingkup deklarasi, parameter prosedur, dan atribut spesifik tipe seperti struktur array atau record. Dalam banyak implementasi compiler, symbol table bersifat hierarkis mengikuti struktur blok dari bahasa pemrograman—sehingga variabel dalam blok lokal akan memiliki entri yang berbeda dari variabel dalam blok global.

Organisasi symbol table dapat dilakukan dengan berbagai cara, seperti hash table, linked list, atau tabel berindeks; namun yang paling umum adalah struktur bertingkat (scoped symbol

table) yang memungkinkan proses lookup mengikuti hierarki lexical scoping. Wirth dalam Pascal-S Implementation Notes menekankan pentingnya pemisahan tipe komposit melalui tabel khusus seperti tabel array atau tabel record, sehingga pengelolaan tipe bersifat modular dan dapat diperluas.

2.4 Lexical Scoping dan Pengaturan Lingkup Identifier

Lexical scoping atau static scoping merupakan mekanisme penentuan lingkup variabel berdasarkan struktur blok dalam kode sumber. Pada model ini, cakupan suatu variabel ditentukan oleh lokasi deklarasinya, bukan oleh urutan eksekusi atau konteks runtime. Ketika compiler memasuki blok baru, ia menciptakan scope baru dalam symbol table, dan ketika keluar, scope tersebut ditutup atau dihapus. Proses lookup identifier dimulai dari scope paling dalam dan bergerak ke luar secara bertahap hingga deklarasi yang sesuai ditemukan.

Model ini memberikan kejelasan referensial dan prediktabilitas dalam penamaan variabel, khas bahasa-bahasa yang dirancang oleh Wirth seperti Pascal, Modula, dan Oberon. Lexical scoping juga memfasilitasi proses semantic analysis karena hubungan antar-blok dapat diwakili secara eksplisit melalui struktur tabel simbol atau daftar display.

2.5 Type System dan Mekanisme Type Checking

Type checking berfungsi memastikan bahwa operasi dalam program dilakukan pada nilai dengan tipe yang sesuai. Pada operasi aritmetika, operand harus bertipe numerik; pada operasi logika, operand harus bertipe boolean; dan assignment hanya dapat dilakukan antara operand dengan tipe yang kompatibel. Dua pendekatan utama type checking adalah static type checking yang dilakukan saat kompilasi, dan dynamic type checking yang dilakukan saat runtime. Pascal dan turunannya adalah bahasa dengan static type system sehingga semua pemeriksaan konsistensi tipe dilakukan selama kompilasi.

Aho et al. (2007) menekankan bahwa type checking merupakan prasyarat penting sebelum generasi kode dilakukan karena kesalahan tipe dapat menghasilkan perilaku tak terdefinisi pada tahap eksekusi. Dalam semantic analysis, setiap node AST dihiasi dengan informasi tipe, sehingga kesalahan seperti type mismatch, pemanggilan fungsi dengan argumen salah, atau operasi ilegal dapat terdeteksi secara sistematis.

2.6 Visitor Pattern sebagai Mekanisme Traversal AST

Visitor pattern adalah pola desain yang memungkinkan pemisahan antara struktur data dan operasi yang diterapkan pada struktur tersebut. Dalam konteks compiler, visitor digunakan untuk melakukan traversal AST dan menerapkan proses semantic analysis tanpa mengubah definisi node AST itu sendiri. Gamma et al. (1994) menyebut visitor sebagai pola yang ideal ketika struktur objek bersifat kompleks, tetap, dan memerlukan berbagai jenis operasi berbeda.

Penerapan visitor dalam semantic analysis memungkinkan compiler memproses setiap node secara sistematis: mengunjungi child node, memperbarui atau membaca symbol table, menghitung atribut semantik seperti tipe ekspresi, dan memberikan anotasi tambahan pada AST. Pendekatan ini juga memudahkan extensibility karena operasi baru dapat ditambahkan tanpa memodifikasi struktur dasar AST.

2.7 Penanganan Kesalahan Semantik (Semantic Error Handling)

Kesalahan semantik terjadi ketika program tidak melanggar sintaks tetapi melanggar aturan makna bahasa. Contoh kesalahan ini meliputi penggunaan identifier yang belum dideklarasikan, deklarasi ganda, ketidaksesuaian tipe pada operasi, pemanggilan fungsi dengan jumlah parameter yang salah, dan indexing array di luar batas. Prinsip utama dalam penanganan kesalahan semantik adalah memberikan pesan kesalahan yang informatif dan mencegah program berhenti secara tiba-tiba.

Compiler modern biasanya mengimplementasikan strategi error recovery, di mana kesalahan dilaporkan tetapi proses analisis tetap berlanjut untuk mendeteksi kesalahan lain. Pendekatan ini meningkatkan pengalaman pengguna dan mempermudah proses debugging karena pengguna dapat melihat semua masalah dalam satu kali kompilasi, bukan satu per satu.

BAB III: Perancangan & Implementasi

3.1 Perancangan

Tahap perancangan pada Milestone 3 bertujuan untuk mendefinisikan arsitektur sistem semantic analysis, struktur modul yang digunakan, serta hubungan antar komponen yang membentuk semantic pipeline pada Pascal-S Compiler. Perancangan ini dilakukan dengan mengacu pada prinsip-prinsip umum compiler construction, di mana proses analisis semantik berada setelah analisis sintaktis dan sebelum tahap code generation. Pada milestone ini, keluaran utama sistem adalah Decorated Abstract Syntax Tree (AST) dan symbol table yang menjadi representasi makna dari program Pascal-S.

Secara keseluruhan, sistem dirancang untuk menerima parse tree yang telah dihasilkan pada Milestone 2, lalu mengubahnya ke bentuk AST melalui syntax-directed translation. AST yang dihasilkan kemudian dilewati oleh semantic visitor yang melakukan verifikasi makna program, termasuk pemeriksaan tipe, validasi lingkup deklarasi, resolusi identifier, serta penambahan anotasi semantik pada setiap node. Proses ini mengacu pada standar semantic analysis dalam literatur compiler dan mengikuti model yang dijelaskan dalam spesifikasi Pascal-S mengenai penggunaan tab, btab, dan atab sebagai struktur penyimpanan simbol.

3.1.1 Arsitektur Sistem

Arsitektur sistem dirancang secara modular dan terlapis (layered architecture) untuk memastikan bahwa proses kompilasi dapat ditelusuri, diuji, dimodifikasi, dan dieksekusi secara terpisah. Pada milestone ini, arsitektur keseluruhan terdiri atas beberapa komponen utama:

Parser & AST Builder

Parser dari milestone sebelumnya menghasilkan *parse tree* sesuai grammar Pascal-S. Pada tahap perancangan ini, parser dikembangkan dengan *semantic actions* yang menerjemahkan node *parse tree* menjadi node AST. Proses ini mengikuti aturan yang didefinisikan dalam *syntax-directed definition*, di mana setiap production rule memiliki aksi pembentukan node AST baru.

Node Representation & AST Structure

Seluruh node AST diimplementasikan sebagai kelas turunan dari ASTNode. Setiap node menyimpan atribut dasar seperti jenis node, posisi dalam kode, dan child nodes. AST menjadi representasi utama yang akan digunakan pada tahap semantic analysis.

Symbol Table System

Sistem tabel simbol mengikuti desain Pascal-S klasik yang terdiri dari tiga tabel: tab untuk seluruh identifier, btab untuk blok prosedur/fungsi, dan atab untuk tipe array. Ketiga tabel bekerja sebagai struktur hirarkis yang menggambarkan lexical scoping dan hubungan antar deklarasi. Desain modular ini memungkinkan pencarian simbol yang konsisten dengan aturan static scoping.

Semantic Visitor

Penggunaan visitor pattern merupakan inti dari tahap ini. Modul visitor menelusuri setiap node AST secara depth-first dan melakukan operasi berikut: memeriksa tipe data untuk operasi aritmetika dan boolean, memvalidasi kompatibilitas assignment, melakukan lookup identifier melalui lexical scope chain, memperbarui tabel simbol ketika memproses deklarasi baru, serta menghiasi node AST dengan tipe hasil evaluasi. Visitor didesain agar setiap fungsi visit_<NodeType> menangani aturan semantik yang spesifik untuk node tersebut.

Error Handler

Modul ini memfasilitasi pelaporan kesalahan semantik seperti identifier yang tidak dideklarasikan, tipe operan yang tidak kompatibel, pengaksesan array di luar batas, dan pemanggilan fungsi dengan parameter salah. Sistem error handling didesain agar proses kompilasi tetap dapat berlanjut tanpa menghentikan eksekusi secara paksa.

Dengan desain modular tersebut, perubahan pada satu komponen tidak memengaruhi komponen lainnya. Misalnya, penambahan dukungan tipe baru pada sistem symbol table tidak mengharuskan perubahan pada struktur AST selama antarmuka yang sama masih digunakan.

3.1.2 Representasi Data dan Struktur AST

Representasi AST dirancang berdasarkan struktur sintaks dan semantik bahasa Pascal-S. Setiap elemen bahasa direpresentasikan sebagai node yang memodelkan operasi atau deklarasi tertentu. Node-node ini diturunkan dari kelas abstrak ASTNode, sehingga mendukung abstraksi dan polimorfisme yang memudahkan ekspansi terhadap fitur bahasa.

AST disusun secara hierarkis dengan beberapa kategori utama. Pada tingkat teratas terdapat node Program dan Block, yang merepresentasikan unit program lengkap beserta konteks deklarasi dan eksekusinya. Di bawahnya terdapat Declaration Nodes yang menangani deklarasi variabel, konstanta, dan tipe. Struktur deklaratif ini penting karena berkaitan langsung dengan pembentukan entri di tabel simbol.

Untuk bagian ekspresi, Expression Nodes seperti BinaryOp, UnaryOp, Literal, dan VarReference memodelkan operasi aritmetika, boolean, serta hubungan antar operand. Node-node ini dirancang agar memiliki kemampuan menyimpan tipe hasil evaluasi, nilai literal,

serta referensi simbol. Seluruh informasi ini akan dihiasi oleh visitor pada tahap analisis semantik.

Sementara itu, Statement Nodes seperti Assignment, IfStatement, WhileStatement, dan ProcedureCall merepresentasikan alur kontrol dan aksi eksekusi program. Desain node statement dibuat konsisten dengan aturan control flow dalam Pascal-S dan memungkinkan visitor menghitung semantic correctness berdasarkan konteks program.

Dengan pembagian seperti ini, AST tidak hanya mencerminkan grammar Pascal-S tetapi juga menjadi representasi semantik lengkap yang siap digunakan untuk tahap kompilasi berikutnya. Struktur AST yang terorganisasi dengan baik memudahkan penerapan semantic rules, pembentukan tabel simbol, serta penambahan anotasi dekoratif yang diperlukan untuk menjalankan fungsi compiler secara konsisten.

3.2 Implementasi

3.2.1 Pembentukan dan Dekorasi AST untuk Analisis Semantik

Tahap implementasi pada milestone ini berfokus pada pembentukan Abstract Syntax Tree (AST) dan pemberian anotasi semantik melalui mekanisme visitor. Setelah parser menghasilkan struktur sintaks awal berdasarkan grammar Pascal-S, AST dibangun dengan menerapkan syntax-directed translation pada setiap production rule. Setiap node AST memiliki struktur kelas sendiri yang merepresentasikan komponen sintaks seperti deklarasi, ekspresi, atau pernyataan. AST ini kemudian menjadi dasar bagi semantic analysis karena memungkinkan traversal struktural yang konsisten dan hirarkis.

Implementasi AST menggunakan pola desain berbasis kelas, di mana seluruh node diturunkan dari kelas dasar ASTNode. Kelas dasar ini menyediakan antarmuka umum seperti accept(visitor) untuk mengaktifkan mekanisme visitor pattern serta to_dict() untuk kebutuhan serialisasi. Desain seperti ini membuat setiap node dapat diproses secara homogen serta memungkinkan penambahan fase analisis tambahan—misalnya code generation—tanpa memodifikasi definisi struktural AST. Pendekatan ini mencerminkan praktik standar dalam konstruksi compiler modern, di mana AST dipisahkan dari logika analisis sehingga perubahan pada salah satu lapisan tidak berdampak langsung pada lapisan lainnya.

3.2.2 Implementasi Program dan Block sebagai Struktur Sintaks Tingkat Atas

Kelas Program dan Block merupakan representasi tingkat tertinggi dalam AST. Program bertindak sebagai akar AST yang memuat identitas program serta blok utama yang berisi deklarasi dan pernyataan. Sementara itu, Block merepresentasikan lexical scope dengan memisahkan deklarasi dari compound statement, sesuai struktur bahasa Pascal yang menggunakan blok bersarang (nested block structure).

Implementasi kedua kelas ini memungkinkan compiler melakukan analisis semantik terhadap deklarasi dan referensi simbol secara terstruktur. Ketika visitor memasuki sebuah Block, lexical level meningkat dan symbol table diperbarui untuk mencerminkan konteks baru. Setelah keluar dari block, lexical level dipulihkan sehingga aturan static scoping dapat diterapkan secara

deterministik. Struktur ini juga memungkinkan semantic analyzer mengidentifikasi kesalahan seperti redeclaration, penggunaan identifier di luar lingkup, atau shadowing yang tidak valid.

```
@dataclass
class Program(ASTNode):
    name: str
    block: ASTNode

    def accept(self, visitor: Any) -> Any:
        return visitor.visit_program(self)

    def to_dict(self) -> dict:
        return {
            'type': 'Program',
            'name': self.name,
            'block': self.block.to_dict()
        }

@dataclass
class Block(ASTNode):
    declarations: list[ASTNode]
    compound_statement: CompoundStatement

    def accept(self, visitor: Any) -> Any:
        return visitor.visit_block(self)

    def to_dict(self) -> dict:
        return {
            'type': 'Block',
            'declarations': [decl.to_dict() for decl in self.declarations],
            'compound_statement': self.compound_statement.to_dict()
        }
```

Struktur ini memperlihatkan pemisahan jelas antara deklarasi dan instruksi, sehingga mekanisme analisis simbol dapat dilakukan secara konsisten mengikuti prinsip lexical scoping yang menjadi dasar semantik Pascal.

3.2.3 Expression

Salah satu komponen kunci dalam analisis semantik adalah symbol table, karena struktur ini menyimpan seluruh informasi mengenai identifier yang digunakan dalam program, seperti variabel, konstanta, prosedur, fungsi, maupun tipe. Pada implementasi ini, desain symbol table mengikuti pendekatan Pascal-S klasik dengan memisahkan entri berdasarkan fungsinya ke dalam tiga tabel berbeda, yaitu tabel identifier (tab), tabel array (atab), dan tabel blok (btab). Pemisahan ini mempermudah pengelolaan tipe komposit dan blok bersarang, serta memungkinkan semantic analyzer melakukan operasi lookup, penyisipan entri baru, dan pelacakan lexical scope secara sistematis. Pada tingkat abstraksi paling dasar, jenis objek yang dapat disimpan dalam tabel dikodekan dalam sebuah enumerasi ObjectKind, sementara setiap entri simbol direpresentasikan oleh kelas SymbolEntry yang menyimpan nama, jenis objek, tipe, serta informasi level dan alamat.

Cuplikan berikut menggambarkan struktur dasar implementasi symbol table beserta entri-entri terkait:

```
# symbol_table.py
from enum import Enum, auto
from typing import Optional
from .types import Type

class ObjectKind(Enum):
    CONSTANT = auto()
    VARIABLE = auto()
    TYPE = auto()
    PROCEDURE = auto()
    FUNCTION = auto()
    PROGRAM = auto()

class SymbolEntry:
    def __init__(
        self,
        name: str,
        obj_kind: ObjectKind,
        type: Type,
        level: int,
        address: int,
        link: Optional[int] = None,
        ref: Optional[int] = None,
    ) -> None:
        self.name = name
        self.obj_kind = obj_kind
        self.type = type
        self.level = level
        self.address = address
        self.link = link           # menghubungkan deklarasi dalam satu blok
        self.ref = ref             # referensi ke atab atau btab untuk tipe
    komposit

    def __repr__(self) -> str:
        return f"<SymbolEntry {self.name} ({self.obj_kind.name})\nlvl={self.level}>"

class ArrayEntry:
    def __init__(
        self,
        index_type: Type,
        low_bound: int,
        high_bound: int,
        element_type: Type,
    ) -> None:
        self.index_type = index_type
        self.low_bound = low_bound
        self.high_bound = high_bound
        self.element_type = element_type
```

```

@property
def size(self) -> int:
    return self.high_bound - self.low_bound + 1


class BlockEntry:
    def __init__(
        self,
        last: Optional[int] = None,
        lpar: Optional[int] = None,
        param_size: int = 0,
        var_size: int = 0,
    ) -> None:
        self.last = last           # indeks identifier terakhir dalam blok ini
        self.lpar = lpar           # pointer ke parameter terakhir
        self.param_size = param_size
        self.var_size = var_size

class SymbolTable:
    def __init__(self) -> None:
        self.tab: list[SymbolEntry] = []
        self.atab: list[ArrayEntry] = []
        self.btab: list[BlockEntry] = []
        self.display: list[int] = []   # display[level] = indeks block active
        self.level: int = -1

    def enter_scope(self) -> None:
        block = BlockEntry()
        self.btab.append(block)
        self.level += 1
        self.display.append(len(self.btab) - 1)

    def leave_scope(self) -> None:
        self.display.pop()
        self.level -= 1

    def insert(
        self,
        name: str,
        obj_kind: ObjectKind,
        type: Type,
        address: int = 0,
        ref: Optional[int] = None,
    ) -> int:
        current_block_index = self.display[self.level]
        current_block = self.btab[current_block_index]
        link = current_block.last

        entry = SymbolEntry(
            name=name,
            obj_kind=obj_kind,
            type=type,
            level=self.level,
            address=address,

```

```

        link=link,
        ref=ref,
    )
    self.tab.append(entry)
    index = len(self.tab) - 1
    current_block.last = index
    return index

def lookup(self, name: str) -> Optional[SymbolEntry]:
    for lvl in reversed(range(self.level + 1)):
        block_index = self.display[lvl]
        entry_index = self.btab[block_index].last
        while entry_index is not None:
            entry = self.tab[entry_index]
            if entry.name == name:
                return entry
            entry_index = entry.link
    return None

def lookup_current_scope(self, name: str) -> Optional[SymbolEntry]:
    block_index = self.display[self.level]
    entry_index = self.btab[block_index].last
    while entry_index is not None:
        entry = self.tab[entry_index]
        if entry.name == name:
            return entry
        entry_index = entry.link
    return None

def add_array_entry(
    self,
    index_type: Type,
    low_bound: int,
    high_bound: int,
    element_type: Type,
) -> int:
    arr = ArrayEntry(index_type, low_bound, high_bound, element_type)
    self.atab.append(arr)
    return len(self.atab) - 1

```

Secara konseptual, cuplikan di atas memperlihatkan bagaimana ruang simbol dibagi dalam beberapa tabel yang saling terkait. Metode enter_scope dan leave_scope mengatur pergantian konteks blok sehingga lexical level dapat dilacak secara eksplisit, sementara metode insert dan lookup bertanggung jawab menambah dan mencari identifier. Pendekatan ini mendukung static scoping karena setiap pencarian dilakukan dari level terdalam ke luar, mengikuti struktur blok bersarang yang menjadi karakteristik bahasa Pascal.

3.2.4 Implementasi Sistem Tipe pada Modul Types

Untuk mendukung pemeriksaan tipe yang konsisten di seluruh tahap analisis semantik, diperlukan representasi tipe yang eksplisit melalui modul types.py. Modul ini mendefinisikan tipe-tipe primitif seperti integer, real, boolean, char, dan string, serta menyediakan abstraksi untuk tipe komposit seperti array atau record. Dengan pendekatan ini, setiap node AST dapat

menyimpan referensi ke objek Type sehingga semantic analyzer dapat membedakan antara operasi aritmetika, logika, dan operasi lain yang bergantung pada tipe operand.

Berikut merupakan cuplikan kode yang menggambarkan struktur dasar sistem tipe:

```
# types.py
from enum import Enum, auto
from dataclasses import dataclass
from typing import Optional


class SimpleTypeKind(Enum):
    INTEGER = auto()
    REAL = auto()
    BOOLEAN = auto()
    CHAR = auto()
    STRING = auto()
    VOID = auto()


@dataclass
class ArrayTypeInfo:
    index_type: "Type"
    low_bound: int
    high_bound: int
    element_type: "Type"


@dataclass
class RecordTypeInfo:
    # Sederhana: representasi record bisa berupa mapping field->Type
    fields: dict[str, "Type"]


@dataclass
class Type:
    kind: SimpleTypeKind
    array_info: Optional[ArrayTypeInfo] = None
    record_info: Optional[RecordTypeInfo] = None

    def is_integer(self) -> bool:
        return self.kind == SimpleTypeKind.INTEGER

    def is_real(self) -> bool:
        return self.kind == SimpleTypeKind.REAL

    def is_boolean(self) -> bool:
        return self.kind == SimpleTypeKind.BOOLEAN

    def is_char(self) -> bool:
        return self.kind == SimpleTypeKind.CHAR

    def is_string(self) -> bool:
        return self.kind == SimpleTypeKind.STRING
```

```

def is_array(self) -> bool:
    return self.array_info is not None

def is_record(self) -> bool:
    return self.record_info is not None

def __repr__(self) -> str:
    if self.is_array():
        return
f"array[{self.array_info.low_bound}..{self.array_info.high_bound}] of
{self.array_info.element_type}"
    if self.is_record():
        return "record"
    return self.kind.name.lower()

INTEGER_TYPE = Type(SimpleTypeKind.INTEGER)
REAL_TYPE = Type(SimpleTypeKind.REAL)
BOOLEAN_TYPE = Type(SimpleTypeKind.BOOLEAN)
CHAR_TYPE = Type(SimpleTypeKind.CHAR)
STRING_TYPE = Type(SimpleTypeKind.STRING)
VOID_TYPE = Type(SimpleTypeKind.VOID)

```

Secara ilmiah, desain tipe seperti di atas mendukung static type checking karena operasi pada AST dapat memanfaatkan metode seperti `is_integer()` atau `is_boolean()` untuk memastikan bahwa operator dan operand kompatibel. Ketika semantic visitor memproses ekspresi biner, ia dapat membandingkan tipe kedua operand dan menentukan tipe hasil, atau sebaliknya melaporkan kesalahan jika ditemukan kombinasi tipe yang tidak sah. Dukungan tipe komposit seperti `ArrayTypeInfo` dan `RecordTypeInfo` juga membuka peluang perluasan bahasa di masa depan tanpa harus merombak keseluruhan sistem.

3.2.5 Implementasi Visitor sebagai Mesin Analisis Semantik

Implementasi visitor pada file `visitor.py` bertindak sebagai mesin utama yang menjalankan analisis semantik di atas struktur AST. Kelas `SemanticVisitor` menyimpan sebuah instance `SymbolTable` dan peta tipe dasar yang memetakan nama tipe pada representasi `Type`. Setiap metode `visit_<NodeType>` didefinisikan untuk menangani aturan semantik spesifik bagi node tersebut, misalnya memproses deklarasi variabel, mengevaluasi tipe ekspresi biner, atau memvalidasi pemanggilan prosedur.

Cuplikan berikut memberikan ilustrasi struktur inti `SemanticVisitor` serta cara ia berinteraksi dengan AST dan symbol table:

```

# visitor.py
from typing import Any
from parse_tree import (
    Program, Block, VarDeclaration, SimpleType,
    AssignmentStatement, BinaryOp, Variable, Number,
)
from .symbol_table import SymbolTable, ObjectKind
from .types import (
    Type, SimpleTypeKind,

```

```

        INTEGER_TYPE, REAL_TYPE, BOOLEAN_TYPE, CHAR_TYPE, STRING_TYPE, VOID_TYPE,
    )

class SemanticVisitor:
    def __init__(self) -> None:
        self.symbol_table = SymbolTable()
        self.type_map: dict[str, Type] = {
            "integer": INTEGER_TYPE,
            "real": REAL_TYPE,
            "boolean": BOOLEAN_TYPE,
            "char": CHAR_TYPE,
            "string": STRING_TYPE,
        }

    def visit(self, node: Any) -> Any:
        method_name = f"visit_{type(node).__name__.lower()}"
        method = getattr(self, method_name, self.generic_visit)
        return method(node)

    def generic_visit(self, node: Any) -> Any:
        raise NotImplementedError(f"No visit_{type(node).__name__.lower()}"
method)

    def visit_program(self, node: Program) -> Any:
        self.symbol_table.enter_scope()
        self.symbol_table.insert(
            name=node.name,
            obj_kind=ObjectKind.PROGRAM,
            type=VOID_TYPE,
        )
        self.visit(node.block)
        self.symbol_table.leave_scope()
        return node

    def visit_block(self, node: Block) -> Any:
        self.symbol_table.enter_scope()
        for decl in node.declarations:
            self.visit(decl)
        self.visit(node.compound_statement)
        self.symbol_table.leave_scope()
        return node

    def visit_vardeclaration(self, node: VarDeclaration) -> Any:
        type_spec = self.visit(node.type_spec)

        for identifier in node.identifiers:
            if self.symbol_table.lookup_current_scope(identifier) is not
None:
                # contoh sederhana: raise error redeclaration
                raise Exception(f"Redeclaration of identifier
'{identifier}'")
            self.symbol_table.insert(
                name=identifier,
                obj_kind=ObjectKind.VARIABLE,
                type=type_spec,

```

```

        )
    return None

def visit_simpletype(self, node: SimpleType) -> Type:
    type_name = node.name.lower()
    if type_name not in self.type_map:
        raise Exception(f"Unknown type '{type_name}'")
    return self.type_map[type_name]

def visit_assignmentstatement(self, node: AssignmentStatement) -> Any:
    var_type = self.visit(node.variable)
    expr_type = self.visit(node.expression)

    if var_type != expr_type:
        raise Exception(
            f"Type mismatch in assignment: {var_type} := {expr_type}"
        )
    return None

def visit_variable(self, node: Variable) -> Type:
    entry = self.symbol_table.lookup(node.name)
    if entry is None:
        raise Exception(f"Undeclared identifier '{node.name}'")
    return entry.type

def visit_number(self, node: Number) -> Type:
    # contoh sederhana: seluruh NUMBER dianggap integer
    return INTEGER_TYPE

def visit_binaryop(self, node: BinaryOp) -> Type:
    left_type = self.visit(node.left)
    right_type = self.visit(node.right)

    if left_type != right_type or not left_type.is_integer():
        raise Exception("Invalid operands for binary operator")
    return left_type

```

Melalui cuplikan tersebut tampak bahwa setiap kunjungan node menghasilkan efek semantik yang spesifik. Metode `visit_program` dan `visit_block` mengelola scope dengan memanggil `enter_scope` dan `leave_scope` pada symbol table, sedangkan `visit_vardeclaration` menyisipkan entri baru ke dalam tabel simbol dan mencegah deklarasi ganda pada scope yang sama. Di sisi lain, metode `visit_assignmentstatement`, `visit_variable`, dan `visit_binaryop` menunjukkan pola pemeriksaan tipe yang konsisten, di mana setiap ekspresi dan operand dikunjungi terlebih dahulu untuk menentukan tipenya sebelum kemudian dilakukan validasi.

BAB IV: Pengujian

Pengujian dilakukan untuk memastikan bahwa sistem yang dibangun—khususnya komponen Lexer dan Abstract Syntax Tree (AST)—telah berfungsi sesuai dengan spesifikasi bahasa Pascal-S yang telah dimodifikasi ke dalam versi berbahasa Indonesia. Secara konseptual, tahap pengujian ini bertujuan untuk memverifikasi keakuratan proses analisis leksikal dalam mengenali dan mengklasifikasikan token, serta memastikan struktur AST yang dihasilkan merepresentasikan hubungan sintaktis dan semantik program secara benar. Selain itu,

pengujian ini juga dimaksudkan untuk menilai ketahanan sistem terhadap kesalahan sintaks, kemampuan penanganan kata kunci majemuk (hyphenated keywords).

4.1 Pengujian Test Cases

Tabel I. SimpleTypesTest

Input
<pre>program SimpleTypes; variabel x, y: integer; a, b: real; flag: boolean; ch: char; mulai x := 10; y := 20; a := 3.14; b := 2.71; flag := true; ch := 'A' selesai.</pre>
Output
<pre>Decorated AST: ===== ProgramNode(name: 'SimpleTypes') └ Block → block_index:0, lev:0 └ Declarations └ VarDecl('x','y') → tab_index:47, type:integer, lev:0 └ VarDecl('a','b') → tab_index:49, type:real, lev:0 └ VarDecl('flag') → tab_index:50, type:boolean, lev:0 └ VarDecl('ch') → tab_index:51, type:char, lev:0 └ CompoundStmt └ Assign → type:void, lev:0 └ Variable('x') → tab_index:46, type:integer, lev:0 └ Number(10) → type:integer └ Assign → type:void, lev:0 └ Variable('y') → tab_index:47, type:integer, lev:0 └ Number(20) → type:integer └ Assign → type:void, lev:0 └ Variable('a') → tab_index:48, type:real, lev:0 └ Number(3.14) → type:real └ Assign → type:void, lev:0 └ Variable('b') → tab_index:49, type:real, lev:0 └ Number(2.71) → type:real └ Assign → type:void, lev:0 └ Variable('flag') → tab_index:50, type:boolean, lev:0 └ Boolean(True) → type:boolean └ Assign → type:void, lev:0 └ Variable('ch') → tab_index:51, type:char, lev:0 └ Char(''A'') → type:char ===== ===== SYMBOL TABLE (tab) ===== Index Name Kind Type Level Addr Ref Link</pre>

0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	SimpleTypes	PROGRAM	void	0	0	0	44
46	x	VARIABLE	integer	0	0	0	0
47	y	VARIABLE	integer	0	1	0	46
48	a	VARIABLE	real	0	2	0	47
49	b	VARIABLE	real	0	3	0	48
50	flag	VARIABLE	boolean	0	4	0	49
51	ch	VARIABLE	char	0	5	0	50

=====

BLOCK TABLE (btab)

=====

Index	Last	LPar	PSize	VSize
0	51	0	0	6

[SUCCESS] Semantic analysis completed without errors.

Bukti dan Keterangan

```
● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test1.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'SimpleTypes')
└ Block → block_index:0, lev:0
    └ Declarations
        └ VarDecl('x','y') → tab_index:47, type:integer, lev:0
        └ VarDecl('a','b') → tab_index:49, type:real, lev:0
        └ VarDecl('flag') → tab_index:50, type:boolean, lev:0
        └ VarDecl('ch') → tab_index:51, type:char, lev:0
    CompoundStmt
        └ Assign → type:void, lev:0
            └ Variable('x') → tab_index:46, type:integer, lev:0
                └ Number(10) → type:integer
            └ Assign → type:void, lev:0
                └ Variable('y') → tab_index:47, type:integer, lev:0
                    └ Number(20) → type:integer
            └ Assign → type:void, lev:0
                └ Variable('a') → tab_index:48, type:real, lev:0
                    └ Number(3.14) → type:real
            └ Assign → type:void, lev:0
                └ Variable('b') → tab_index:49, type:real, lev:0
                    └ Number(2.71) → type:real
            └ Assign → type:void, lev:0
                └ Variable('flag') → tab_index:50, type:boolean, lev:0
                    └ Boolean(True) → type:boolean
            └ Assign → type:void, lev:0
                └ Variable('ch') → tab_index:51, type:char, lev:0
                    └ Char('A') → type:char
=====
=====

SYMBOL TABLE (tab)
=====
Index Name Kind Type Level Addr Ref Link
-----
0 salah CONSTANT boolean 0 0 0 0
1 benar CONSTANT boolean 0 0 0 0
2 integer TYPE integer 0 0 0 1
3 real TYPE real 0 0 0 2
4 boolean TYPE boolean 0 0 0 3
5 char TYPE char 0 0 0 4
```

```

6   string      TYPE     string      0   0   0   5
7   dan        PROCEDURE void      0   0   0   6
8   atau       PROCEDURE void      0   0   0   7
9   tidak      PROCEDURE void      0   0   0   8
10  bagi       PROCEDURE void      0   0   0   9
11  mod        PROCEDURE void      0   0   0   10
12  program    PROCEDURE void      0   0   0   11
13  variabel   PROCEDURE void      0   0   0   12
14  konstanta PROCEDURE void      0   0   0   13
15  tipe        PROCEDURE void      0   0   0   14
16  prosedur   PROCEDURE void      0   0   0   15
17  fungsi     PROCEDURE void      0   0   0   16
18  mulai      PROCEDURE void      0   0   0   17
19  selesai    PROCEDURE void      0   0   0   18
20  jika        PROCEDURE void      0   0   0   19
21  maka        PROCEDURE void      0   0   0   20
22  selain-itu PROCEDURE void      0   0   0   21
23  selama     PROCEDURE void      0   0   0   22
24  lakukan   PROCEDURE void      0   0   0   23
25  untuk      PROCEDURE void      0   0   0   24
26  ke          PROCEDURE void      0   0   0   25
27  turun-ke   PROCEDURE void      0   0   0   26
28  larik      PROCEDURE void      0   0   0   27
29  write       PROCEDURE void      0   0   0   28
30  writeln    PROCEDURE void      0   0   0   29
31  read        PROCEDURE void      0   0   0   30
32  readln     PROCEDURE void      0   0   0   31
33  abs         FUNCTION integer  0   0   0   32
34  sqr         FUNCTION integer  0   0   0   33
35  sqrt        FUNCTION real    0   0   0   34
36  sin         FUNCTION real    0   0   0   35
37  cos         FUNCTION real    0   0   0   36
38  exp         FUNCTION real    0   0   0   37
39  ln          FUNCTION real    0   0   0   38
40  odd         FUNCTION boolean 0   0   0   39
41  ord         FUNCTION integer 0   0   0   40
42  chr         FUNCTION char   0   0   0   41
43  succ        FUNCTION integer 0   0   0   42
44  pred        FUNCTION integer 0   0   0   43
45  SimpleTypes PROGRAM void    0   0   0   44
46  x           VARIABLE integer 0   0   0   45
47  y           VARIABLE integer 0   1   0   46
48  a           VARIABLE real   0   2   0   47
49  b           VARIABLE real   0   3   0   48

50  flag        VARIABLE boolean 0   4   0   49
51  ch          VARIABLE char   0   5   0   50

=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----
0      51    0      0       6

[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani simple type pada assignment ke variabel, hasil uji sudah benar.

Tabel II. TypeCheckingTest

Input

```

program TypeChecking;
variabel
  x, y, z: integer;
  result: real;
mulai
  x := 10;
  y := 20;
  z := x + y;
  result := x + y;
  result := result * 2.5
selesai.

```

Output

Decorated AST:							
=====							
ProgramNode(name: 'TypeChecking')							
└ Block → block_index:0, lev:0							
└ Declarations							
└ VarDecl('x','y','z') → tab_index:48, type:integer, lev:0							
└ VarDecl('result') → tab_index:49, type:real, lev:0							
└ CompoundStmt							
└ Assign → type:void, lev:0							
└ Variable('x') → tab_index:46, type:integer, lev:0							
└ Number(10) → type:integer							
└ Assign → type:void, lev:0							
└ Variable('y') → tab_index:47, type:integer, lev:0							
└ Number(20) → type:integer							
└ Assign → type:void, lev:0							
└ Variable('z') → tab_index:48, type:integer, lev:0							
└ BinOp '+' → type:integer							
└ Variable('x') → tab_index:46, type:integer, lev:0							
└ Variable('y') → tab_index:47, type:integer, lev:0							
└ Assign → type:void, lev:0							
└ Variable('result') → tab_index:49, type:real, lev:0							
└ BinOp '*' → type:real							
└ Variable('result') → tab_index:49, type:real, lev:0							
└ Number(2.5) → type:real							
└ Assign → type:void, lev:0							
└ Variable('result') → tab_index:49, type:real, lev:0							
└ BinOp '*' → type:real							
└ Variable('result') → tab_index:49, type:real, lev:0							
└ Number(2.5) → type:real							
=====							
=====							
SYMBOL TABLE (tab)							
=====							
Index	Name	Kind	Type	Level	Addr	Ref	Link
-----	-----	-----	-----	-----	-----	-----	-----
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22

```

24    lakukan      PROCEDURE void      0      0      0      23
25    untuk        PROCEDURE void      0      0      0      24
26    ke           PROCEDURE void      0      0      0      25
27    turun-ke     PROCEDURE void      0      0      0      26
28    larik        PROCEDURE void      0      0      0      27
29    write         PROCEDURE void      0      0      0      28
30    writeln      PROCEDURE void      0      0      0      29
31    read          PROCEDURE void      0      0      0      30
32    readln       PROCEDURE void      0      0      0      31
33    abs           FUNCTION integer   0      0      0      32
34    sqr           FUNCTION integer   0      0      0      33
35    sqrt          FUNCTION real     0      0      0      34
36    sin           FUNCTION real     0      0      0      35
37    cos           FUNCTION real     0      0      0      36
38    exp           FUNCTION real     0      0      0      37
39    ln            FUNCTION real     0      0      0      38
40    odd           FUNCTION boolean  0      0      0      39
41    ord           FUNCTION integer  0      0      0      40
42    chr           FUNCTION char    0      0      0      41
43    succ          FUNCTION integer  0      0      0      42
44    pred          FUNCTION integer  0      0      0      43
45    TypeChecking PROGRAM void     0      0      0      44
46    x              VARIABLE integer  0      0      0      0
47    y              VARIABLE integer  0      1      0      46
48    z              VARIABLE integer  0      2      0      47
49    result         VARIABLE real    0      3      0      48

=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----
0      49     0      0      4

[SUCCESS] Semantic analysis completed without errors.

```

Bukti dan Keterangan

```

PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-6) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/
● test2.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'TypeChecking')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('x','y','z') → tab_index:48, type:integer, lev:0
    └ VarDecl('result') → tab_index:49, type:real, lev:0
  CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('x') → tab_index:46, type:integer, lev:0
      └ Number(18) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('y') → tab_index:47, type:integer, lev:0
      └ Number(20) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('z') → tab_index:48, type:integer, lev:0
      └ BinOp '+' → type:integer
        └ Variable('x') → tab_index:46, type:integer, lev:0
        └ Variable('y') → tab_index:47, type:integer, lev:0
    └ Assign → type:void, lev:0
      └ Variable('result') → tab_index:49, type:real, lev:0
      └ BinOp '*' → type:real
        └ Variable('x') → tab_index:46, type:integer, lev:0
        └ Variable('y') → tab_index:47, type:integer, lev:0
    └ Assign → type:void, lev:0
      └ Variable('result') → tab_index:49, type:real, lev:0
      └ BinOp '*' → type:real
        └ Variable('result') → tab_index:49, type:real, lev:0
        └ Number(2.5) → type:real
=====

=====
SYMBOL TABLE (tab)
=====

```

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3

```

5   char          TYPE    char      0   0   0   4
6   string         TYPE    string    0   0   0   5
7   dan           PROCEDURE void     0   0   0   6
8   atau          PROCEDURE void     0   0   0   7
9   tidak          PROCEDURE void     0   0   0   8
10  bagi           PROCEDURE void     0   0   0   9
11  mod            PROCEDURE void     0   0   0   10
12  program        PROCEDURE void    0   0   0   11
13  variabel       PROCEDURE void    0   0   0   12
14  konstanta     PROCEDURE void    0   0   0   13
15  tipe           PROCEDURE void    0   0   0   14
16  prosedur       PROCEDURE void    0   0   0   15
17  fungsi         PROCEDURE void    0   0   0   16
18  mulai          PROCEDURE void    0   0   0   17
19  selesai        PROCEDURE void    0   0   0   18
20  jika           PROCEDURE void    0   0   0   19
21  maka           PROCEDURE void    0   0   0   20
22  selain-itu    PROCEDURE void    0   0   0   21
23  selama         PROCEDURE void    0   0   0   22
24  lakukan       PROCEDURE void    0   0   0   23
25  untuk          PROCEDURE void    0   0   0   24
26  ke              PROCEDURE void    0   0   0   25
27  turun-ke      PROCEDURE void    0   0   0   26
28  larik          PROCEDURE void    0   0   0   27
29  write          PROCEDURE void    0   0   0   28
30  writeln        PROCEDURE void    0   0   0   29
31  read           PROCEDURE void    0   0   0   30
32  readln         PROCEDURE void    0   0   0   31
33  abs             FUNCTION integer 0   0   0   32
34  sqr             FUNCTION integer 0   0   0   33
35  sqrt            FUNCTION real    0   0   0   34
36  sin              FUNCTION real    0   0   0   35
37  cos              FUNCTION real    0   0   0   36
38  exp              FUNCTION real    0   0   0   37
39  ln               FUNCTION real    0   0   0   38
40  odd              FUNCTION boolean 0   0   0   39
41  ord              FUNCTION integer 0   0   0   40
42  chr              FUNCTION char    0   0   0   41
43  succ             FUNCTION integer 0   0   0   42
44  pred             FUNCTION integer 0   0   0   43
45  TypeChecking    PROGRAM void     0   0   0   44
46  x                VARIABLE integer 0   0   0   0
47  y                VARIABLE integer 0   1   0   46
48  z                VARIABLE integer 0   2   0   47

49  result          VARIABLE real    0   3   0   48

=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----+
0      49     0      0       4

[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani type checking pada operasi dan assignment ke variabel, hasil uji sudah benar.

Tabel III. ScopeBasicTest

Input
<pre> program ScopeBasic; variabel x: integer; prosedur TestProc; variabel y: integer; mulai y := 10; x := 20 selesai; mulai </pre>

```

x := 5;
TestProc()
selesai.

```

Output

Decorated AST:

```

=====
ProgramNode(name: 'ScopeBasic')
└ Block → block_index:0, lev:0
    └ Declarations
        └ VarDecl('x') → tab_index:46, type:integer, lev:0
        └ ProcDecl('TestProc')
            └ Block → block_index:1, lev:1
                └ Declarations
                    └ VarDecl('y') → tab_index:48, type:integer, lev:1
                └ CompoundStmt
                    └ Assign → type:void, lev:1
                        └ Variable('y') → tab_index:48, type:integer, lev:1
                            └ Number(10) → type:integer
                    └ Assign → type:void, lev:1
                        └ Variable('x') → tab_index:46, type:integer, lev:0
                            └ Number(20) → type:integer
                └ CompoundStmt
                    └ Assign → type:void, lev:0
                        └ Variable('x') → tab_index:46, type:integer, lev:0
                            └ Number(5) → type:integer
                    └ ProcCall('TestProc') → tab_index:47, type:void, lev:0
=====
```

=====
SYMBOL TABLE (tab)
=====

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24

```

26   ke          PROCEDURE void      0  0  0  25
27   turun-ke    PROCEDURE void      0  0  0  26
28   larik       PROCEDURE void      0  0  0  27
29   write        PROCEDURE void      0  0  0  28
30   writeln     PROCEDURE void      0  0  0  29
31   read         PROCEDURE void      0  0  0  30
32   readln      PROCEDURE void      0  0  0  31
33   abs          FUNCTION integer  0  0  0  32
34   sqr          FUNCTION integer  0  0  0  33
35   sqrt         FUNCTION real    0  0  0  34
36   sin          FUNCTION real    0  0  0  35
37   cos          FUNCTION real    0  0  0  36
38   exp          FUNCTION real    0  0  0  37
39   ln           FUNCTION real    0  0  0  38
40   odd          FUNCTION boolean 0  0  0  39
41   ord          FUNCTION integer 0  0  0  40
42   chr          FUNCTION char   0  0  0  41
43   succ         FUNCTION integer 0  0  0  42
44   pred         FUNCTION integer 0  0  0  43
45   ScopeBasic   PROGRAM void    0  0  0  44
46   x            VARIABLE integer 0  0  0  0
47   TestProc     PROCEDURE void    0  0  1  46
48   y            VARIABLE integer 1  3  0  0

=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----
0      47     0      0       1
1      48     0      0       1

[SUCCESS] Semantic analysis completed without errors.

```

Bukti dan Keterangan

```

● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/
test3.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'ScopeBasic')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('x') → tab_index:46, type:integer, lev:0
    └ ProcDecl('TestProc')
      └ Block → block_index:1, lev:1
        └ Declarations
          └ VarDecl('y') → tab_index:48, type:integer, lev:1
        └ CompoundStmt
          └ Assign → type:void, lev:1
            └ Variable('y') → tab_index:48, type:integer, lev:1
            └ Number(10) → type:integer
          └ Assign → type:void, lev:1
            └ Variable('x') → tab_index:46, type:integer, lev:0
            └ Number(20) → type:integer
        └ CompoundStmt
          └ Assign → type:void, lev:0
            └ Variable('x') → tab_index:46, type:integer, lev:0
            └ Number(5) → type:integer
          └ ProcCall('TestProc') → tab_index:47, type:void, lev:0
=====

=====
SYMBOL TABLE (tab)
=====

```

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	ScopeBasic	PROGRAM	void	0	0	0	44
46	x	VARIABLE	integer	0	0	0	0
47	TestProc	PROCEDURE	void	0	0	1	46
48	y	VARIABLE	integer	1	3	0	0

```

=====
BLOCK TABLE (btab)
=====
Index Last LPar PSize VSize
-----
0    47   0    0    1
1    48   0    0    1
=====
[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani operasi scope basic, hasil uji sudah benar.

Tabel IV. NestedScopeTest

Input
<pre> program NestedScope; variabel global_x: integer; prosedur Outer; variabel outer_y: integer; prosedur Inner; variabel inner_z: integer; mulai inner_z := 1; outer_y := 2; global_x := 3 selesai; mulai outer_y := 10; Inner() selesai; mulai global_x := 100; Outer() selesai. </pre>
Output
<pre> Decorated AST: ===== ProgramNode(name: 'NestedScope') └ Block → block_index:0, lev:0 └ Declarations └ VarDecl('global_x') → tab_index:46, type:integer, lev:0 └ ProcDecl('Outer') └ Block → block_index:1, lev:1 └ Declarations └ VarDecl('outer_y') → tab_index:48, type:integer, lev:1 └ ProcDecl('Inner') └ Block → block_index:2, lev:2 └ Declarations └ VarDecl('inner_z') → tab_index:50, type:integer, lev:2 └ CompoundStmt </pre>

```

    └─ Assign → type:void, lev:2
      └─ Variable('inner_z') → tab_index:50, type:integer, lev:2
        └─ Number(1) → type:integer
    └─ Assign → type:void, lev:2
      └─ Variable('outer_y') → tab_index:48, type:integer, lev:1
        └─ Number(2) → type:integer
    └─ Assign → type:void, lev:2
      └─ Variable('global_x') → tab_index:46, type:integer, lev:0
        └─ Number(3) → type:integer
  CompoundStmt
  └─ Assign → type:void, lev:1
    └─ Variable('outer_y') → tab_index:48, type:integer, lev:1
      └─ Number(10) → type:integer
  ProcCall('Inner') → tab_index:49, type:void, lev:1
  =====
  CompoundStmt
  └─ Assign → type:void, lev:0
    └─ Variable('global_x') → tab_index:46, type:integer, lev:0
      └─ Number(100) → type:integer
  ProcCall('Outer') → tab_index:47, type:void, lev:0
=====

=====
SYMBOL TABLE (tab)
=====

```

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42

```

44     pred           FUNCTION    integer   0   0   0   43
45     NestedScope  PROGRAM     void      0   0   0   44
46     global_x     VARIABLE   integer   0   0   0   0
47     Outer        PROCEDURE  void      0   0   1   46
48     outer_y      VARIABLE   integer   1   3   0   0
49     Inner        PROCEDURE  void      1   0   2   48
50     inner_z      VARIABLE   integer   2   3   0   0

=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----
0      47     0      0       1
1      49     0      0       1
2      50     0      0       1

[SUCCESS] Semantic analysis completed without errors.

```

Bukti dan Keterangan

```

● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test4.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'NestedScope')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('global_x') → tab_index:46, type:integer, lev:0
  └ ProcDecl('Outer')
    └ Block → block_index:1, lev:1
      └ Declarations
        └ VarDecl('outer_y') → tab_index:48, type:integer, lev:1
        └ ProcDecl('Inner')
          └ Block → block_index:2, lev:2
            └ Declarations
              └ VarDecl('inner_z') → tab_index:50, type:integer, lev:2
            └ CompoundStmt
              └ Assign → type:void, lev:2
                └ Variable('inner_z') → tab_index:50, type:integer, lev:2
                └ Number(1) → type:integer
              └ Assign → type:void, lev:2
                └ Variable('outer_y') → tab_index:48, type:integer, lev:1
                └ Number(2) → type:integer
              └ Assign → type:void, lev:2
                └ Variable('global_x') → tab_index:46, type:integer, lev:0
                └ Number(3) → type:integer
            └ CompoundStmt
              └ Assign → type:void, lev:1
                └ Variable('outer_y') → tab_index:48, type:integer, lev:1
                └ Number(10) → type:integer
              └ ProcCall('Inner') → tab_index:49, type:void, lev:1
            └ CompoundStmt
              └ Assign → type:void, lev:0
                └ Variable('global_x') → tab_index:46, type:integer, lev:0
                └ Number(100) → type:integer
              └ ProcCall('Outer') → tab_index:47, type:void, lev:0
  =====
SYMBOL TABLE (tab)
=====
Index  Name   Kind   Type   Level  Addr  Ref  Link
-----
```

```

0  salah      CONSTANT boolean   0  0  0  0
1  benar      CONSTANT boolean   0  0  0  0
2  integer    TYPE    integer   0  0  0  1
3  real       TYPE    real     0  0  0  2
4  boolean    TYPE    boolean  0  0  0  3
5  char       TYPE    char    0  0  0  4
6  string     TYPE    string  0  0  0  5
7  dan        PROCEDURE void    0  0  0  6
8  atau       PROCEDURE void    0  0  0  7
9  tidak      PROCEDURE void    0  0  0  8
10 bagl       PROCEDURE void   0  0  0  9
11 mod        PROCEDURE void   0  0  0 10
12 program   PROCEDURE void   0  0  0 11
13 variabel   PROCEDURE void   0  0  0 12
14 konstanta PROCEDURE void   0  0  0 13
15 tipe       PROCEDURE void   0  0  0 14
16 prosedur   PROCEDURE void   0  0  0 15
17 fungsi    PROCEDURE void   0  0  0 16
18 mulai      PROCEDURE void   0  0  0 17
19 selesai   PROCEDURE void   0  0  0 18
20 jika       PROCEDURE void   0  0  0 19
21 maka       PROCEDURE void   0  0  0 20
22 selain-itu PROCEDURE void   0  0  0 21
23 selama    PROCEDURE void   0  0  0 22
24 lakukan   PROCEDURE void   0  0  0 23
25 untuk     PROCEDURE void   0  0  0 24
26 ke         PROCEDURE void   0  0  0 25
27 turun-ke  PROCEDURE void   0  0  0 26
28 larik     PROCEDURE void   0  0  0 27
29 write      PROCEDURE void   0  0  0 28
30 writeln   PROCEDURE void   0  0  0 29
31 read      PROCEDURE void   0  0  0 30
32 readln   PROCEDURE void   0  0  0 31
33 abs       FUNCTION integer 0  0  0 32
34 sqr       FUNCTION integer 0  0  0 33
35 sqrt      FUNCTION real   0  0  0 34
36 sin       FUNCTION real   0  0  0 35
37 cos       FUNCTION real   0  0  0 36
38 exp       FUNCTION real   0  0  0 37
39 ln        FUNCTION real   0  0  0 38
40 odd       FUNCTION boolean 0  0  0 39
41 ord        FUNCTION integer 0  0  0 40
42 chr        FUNCTION char   0  0  0 41
43 succ       FUNCTION integer 0  0  0 42

44 pred       FUNCTION integer 0  0  0 43
45 NestedScope PROGRAM void    0  0  0 44
46 global_x   VARIABLE integer 0  0  0  0
47 Outer      PROCEDURE void   0  0  1 46
48 outer_y   VARIABLE integer 1  3  0  0
49 Inner      PROCEDURE void   1  0  2 48
50 inner_z   VARIABLE integer 2  3  0  0

=====
BLOCK TABLE (btab)
=====
Index Last  LPar  PSize  VSize
-----
0    47    0    0    1
1    49    0    0    1
2    50    0    0    1

[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani operasi nested scope, hasil uji sudah benar.

Tabel V. ArrayBasicTest

Input				
<pre> program ArrayBasic; variabel arr: larik[1..10] dari integer; i: integer; mulai i := 1; arr[i] := 100; arr[5] := 200; </pre>				

```
i := arr[5]
selesai.
```

Output

Decorated AST:

```
=====
ProgramNode(name: 'ArrayBasic')
└ Block → block_index:0, lev:0
    └ Declarations
        └ VarDecl('arr') → tab_index:46, type:array of integer, lev:0
        └ VarDecl('i') → tab_index:47, type:integer, lev:0
    CompoundStmt
        └ Assign → type:void, lev:0
            └ Variable('i') → tab_index:47, type:integer, lev:0
            └ Number(1) → type:integer
        └ Assign → type:void, lev:0
            └ VarAccess('arr'[]) → tab_index:46, type:integer, lev:0
                └ Variable('i') → tab_index:47, type:integer, lev:0
            └ Number(100) → type:integer
        └ Assign → type:void, lev:0
            └ VarAccess('arr'[]) → tab_index:46, type:integer, lev:0
                └ Number(5) → type:integer
            └ Number(200) → type:integer
        └ Assign → type:void, lev:0
            └ Variable('i') → tab_index:47, type:integer, lev:0
            └ VarAccess('arr'[]) → tab_index:46, type:integer, lev:0
                └ Number(5) → type:integer
=====
```

=====
SYMBOL TABLE (tab)
=====

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24

```

26   ke          PROCEDURE void      0    0    0    25
27   turun-ke   PROCEDURE void      0    0    0    26
28   larik       PROCEDURE void      0    0    0    27
29   write        PROCEDURE void      0    0    0    28
30   writeln     PROCEDURE void      0    0    0    29
31   read         PROCEDURE void      0    0    0    30
32   readln      PROCEDURE void      0    0    0    31
33   abs          FUNCTION integer  0    0    0    32
34   sqr          FUNCTION integer  0    0    0    33
35   sqrt         FUNCTION real    0    0    0    34
36   sin          FUNCTION real    0    0    0    35
37   cos          FUNCTION real    0    0    0    36
38   exp          FUNCTION real    0    0    0    37
39   ln           FUNCTION real    0    0    0    38
40   odd          FUNCTION boolean 0    0    0    39
41   ord          FUNCTION integer 0    0    0    40
42   chr          FUNCTION char   0    0    0    41
43   succ         FUNCTION integer 0    0    0    42
44   pred         FUNCTION integer 0    0    0    43
45   ArrayBasic   PROGRAM void    0    0    0    44
46   arr          VARIABLE array of integer 0    0    0
47   i             VARIABLE integer 0    1    0    46

=====
ARRAY TABLE (atab)
=====
Index  IdxType   ElemType   Low   High   Elemsz   Size
-----
0      integer    integer    1     10     1       10

=====
BLOCK TABLE (btab)
=====
Index  Last    LPar    PSize   VSize
-----
0      47     0      0       2

[SUCCESS] Semantic analysis completed without errors.

```

Bukti dan Keterangan

```

● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/
test5.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'ArrayBasic')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('arr') → tab_index:46, type:array of integer, lev:0
    └ VarDecl('i') → tab_index:47, type:integer, lev:0
  CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('i') → tab_index:47, type:integer, lev:0
      └ Number(1) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('arr[]') → tab_index:46, type:integer, lev:0
        └ Variable('i') → tab_index:47, type:integer, lev:0
        └ Number(100) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('arr[]') → tab_index:46, type:integer, lev:0
        └ Number(5) → type:integer
        └ Number(200) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('arr[]') → tab_index:46, type:integer, lev:0
        └ Number(5) → type:integer
=====

=====
SYMBOL TABLE (tab)
=====


| Index | Name       | Kind      | Type             | Level | Addr | Ref | Link |
|-------|------------|-----------|------------------|-------|------|-----|------|
| 0     | salah      | CONSTANT  | boolean          | 0     | 0    | 0   | 0    |
| 1     | benar      | CONSTANT  | boolean          | 0     | 0    | 0   | 0    |
| 2     | integer    | TYPE      | integer          | 0     | 0    | 0   | 1    |
| 3     | real       | TYPE      | real             | 0     | 0    | 0   | 2    |
| 4     | boolean    | TYPE      | boolean          | 0     | 0    | 0   | 3    |
| 5     | char       | TYPE      | char             | 0     | 0    | 0   | 4    |
| 6     | string     | TYPE      | string           | 0     | 0    | 0   | 5    |
| 7     | dan        | PROCEDURE | void             | 0     | 0    | 0   | 6    |
| 8     | atau       | PROCEDURE | void             | 0     | 0    | 0   | 7    |
| 9     | tidak      | PROCEDURE | void             | 0     | 0    | 0   | 8    |
| 10    | bagi       | PROCEDURE | void             | 0     | 0    | 0   | 9    |
| 11    | mod        | PROCEDURE | void             | 0     | 0    | 0   | 10   |
| 12    | program    | PROCEDURE | void             | 0     | 0    | 0   | 11   |
| 13    | variabel   | PROCEDURE | void             | 0     | 0    | 0   | 12   |
| 14    | konstanta  | PROCEDURE | void             | 0     | 0    | 0   | 13   |
| 15    | tipe       | PROCEDURE | void             | 0     | 0    | 0   | 14   |
| 16    | prosedur   | PROCEDURE | void             | 0     | 0    | 0   | 15   |
| 17    | fungsi     | PROCEDURE | void             | 0     | 0    | 0   | 16   |
| 18    | mulai      | PROCEDURE | void             | 0     | 0    | 0   | 17   |
| 19    | selesai    | PROCEDURE | void             | 0     | 0    | 0   | 18   |
| 20    | jika       | PROCEDURE | void             | 0     | 0    | 0   | 19   |
| 21    | maka       | PROCEDURE | void             | 0     | 0    | 0   | 20   |
| 22    | selain-itu | PROCEDURE | void             | 0     | 0    | 0   | 21   |
| 23    | selama     | PROCEDURE | void             | 0     | 0    | 0   | 22   |
| 24    | lakukan    | PROCEDURE | void             | 0     | 0    | 0   | 23   |
| 25    | untuk      | PROCEDURE | void             | 0     | 0    | 0   | 24   |
| 26    | ke         | PROCEDURE | void             | 0     | 0    | 0   | 25   |
| 27    | turun-ke   | PROCEDURE | void             | 0     | 0    | 0   | 26   |
| 28    | larik      | PROCEDURE | void             | 0     | 0    | 0   | 27   |
| 29    | write      | PROCEDURE | void             | 0     | 0    | 0   | 28   |
| 30    | writeln    | PROCEDURE | void             | 0     | 0    | 0   | 29   |
| 31    | read       | PROCEDURE | void             | 0     | 0    | 0   | 30   |
| 32    | readln     | PROCEDURE | void             | 0     | 0    | 0   | 31   |
| 33    | abs        | FUNCTION  | integer          | 0     | 0    | 0   | 32   |
| 34    | sqr        | FUNCTION  | integer          | 0     | 0    | 0   | 33   |
| 35    | sqrt       | FUNCTION  | real             | 0     | 0    | 0   | 34   |
| 36    | sin        | FUNCTION  | real             | 0     | 0    | 0   | 35   |
| 37    | cos        | FUNCTION  | real             | 0     | 0    | 0   | 36   |
| 38    | exp        | FUNCTION  | real             | 0     | 0    | 0   | 37   |
| 39    | ln         | FUNCTION  | real             | 0     | 0    | 0   | 38   |
| 40    | odd        | FUNCTION  | boolean          | 0     | 0    | 0   | 39   |
| 41    | ord        | FUNCTION  | integer          | 0     | 0    | 0   | 40   |
| 42    | chr        | FUNCTION  | char             | 0     | 0    | 0   | 41   |
| 43    | succ       | FUNCTION  | integer          | 0     | 0    | 0   | 42   |
| 44    | pred       | FUNCTION  | integer          | 0     | 0    | 0   | 43   |
| 45    | ArrayBasic | PROGRAM   | void             | 0     | 0    | 0   | 44   |
| 46    | arr        | VARIABLE  | array of integer | 0     | 0    | 0   | 0    |
| 47    | i          | VARIABLE  | integer          | 0     | 1    | 0   | 46   |


```

```

=====
ARRAY TABLE (atab)
=====
Index  IdxType     ElemType   Low    High   Elemsz  Size
-----  -----      -----      ---    ---    ---    ---
0      integer     integer    1      10     1      10

=====
BLOCK TABLE (btab)
=====
Index  Last       LPar      PSize   VSize
-----  ---       ---      ---    ---
0      47        0        0       2

[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani tipe data array basic, hasil uji sudah benar.

Tabel VI. RecordBasicTest

Input
<pre> program RecordBasic; tipe Person = rekaman age: integer; height: real; selesai; variabel p: Person; mulai p.age := 25; p.height := 175.5 selesai. </pre>
Output
<pre> Decorated AST: ===== ProgramNode(name: 'RecordBasic') └ Block → block_index:0, lev:0 └ Declarations └ TypeDecl('Person') └ RecordType └ VarDecl('age') └ VarDecl('height') └ VarDecl('p') → tab_index:47, type:record, lev:0 CompoundStmt └ Assign → type:void, lev:0 └ VarAccess('.age') → tab_index:47, type:integer, lev:0 └ Number(25) → type:integer └ Assign → type:void, lev:0 └ VarAccess('.height') → tab_index:47, type:real, lev:0 └ Number(175.5) → type:real ===== ===== SYMBOL TABLE (tab) ===== Index Name Kind Type Level Addr Ref Link ----- --- --- --- --- --- --- --- 0 salah CONSTANT boolean 0 0 0 0 </pre>

1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	RecordBasic	PROGRAM	void	0	0	0	44
46	Person	TYPE	record	0	0	0	45
47	p	VARIABLE	record	0	0	0	0
<hr/>							
BLOCK TABLE (btab)							
<hr/>							
Index	Last	LPar	PSize	VSize			
0	47	0	0	1			
[SUCCESS] Semantic analysis completed without errors.							
Bukti dan Keterangan							

```

PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test6.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'RecordBasic')
└ Block → block_index:0, lev:0
  └ Declarations
    └ TypeDecl('Person')
      └ RecordType
        └ VarDecl('age')
        └ VarDecl('height')
    └ VarDecl('p') → tab_index:47, type:record, lev:0
  CompoundStmt
    └ Assign → type:void, lev:0
      └ VarAccess('.age') → tab_index:47, type:integer, lev:0
      └ Number(25) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('.height') → tab_index:47, type:real, lev:0
      └ Number(175.5) → type:real
=====

=====
SYMBOL TABLE (tab)
=====


| Index | Name        | Kind      | Type    | Level | Addr | Ref | Link |
|-------|-------------|-----------|---------|-------|------|-----|------|
| 0     | salah       | CONSTANT  | boolean | 0     | 0    | 0   | 0    |
| 1     | benar       | CONSTANT  | boolean | 0     | 0    | 0   | 0    |
| 2     | integer     | TYPE      | integer | 0     | 0    | 0   | 1    |
| 3     | real        | TYPE      | real    | 0     | 0    | 0   | 2    |
| 4     | boolean     | TYPE      | boolean | 0     | 0    | 0   | 3    |
| 5     | char        | TYPE      | char    | 0     | 0    | 0   | 4    |
| 6     | string      | TYPE      | string  | 0     | 0    | 0   | 5    |
| 7     | dan         | PROCEDURE | void    | 0     | 0    | 0   | 6    |
| 8     | atau        | PROCEDURE | void    | 0     | 0    | 0   | 7    |
| 9     | tidak       | PROCEDURE | void    | 0     | 0    | 0   | 8    |
| 10    | bagi        | PROCEDURE | void    | 0     | 0    | 0   | 9    |
| 11    | mod         | PROCEDURE | void    | 0     | 0    | 0   | 10   |
| 12    | program     | PROCEDURE | void    | 0     | 0    | 0   | 11   |
| 13    | variabel    | PROCEDURE | void    | 0     | 0    | 0   | 12   |
| 14    | konstanta   | PROCEDURE | void    | 0     | 0    | 0   | 13   |
| 15    | tipe        | PROCEDURE | void    | 0     | 0    | 0   | 14   |
| 16    | prosedur    | PROCEDURE | void    | 0     | 0    | 0   | 15   |
| 17    | fungsi      | PROCEDURE | void    | 0     | 0    | 0   | 16   |
| 18    | mula        | PROCEDURE | void    | 0     | 0    | 0   | 17   |
| 19    | selesai     | PROCEDURE | void    | 0     | 0    | 0   | 18   |
| 20    | jika        | PROCEDURE | void    | 0     | 0    | 0   | 19   |
| 21    | maka        | PROCEDURE | void    | 0     | 0    | 0   | 20   |
| 22    | selain-itu  | PROCEDURE | void    | 0     | 0    | 0   | 21   |
| 23    | selama      | PROCEDURE | void    | 0     | 0    | 0   | 22   |
| 24    | lakukan     | PROCEDURE | void    | 0     | 0    | 0   | 23   |
| 25    | untuk       | PROCEDURE | void    | 0     | 0    | 0   | 24   |
| 26    | ke          | PROCEDURE | void    | 0     | 0    | 0   | 25   |
| 27    | turun-ke    | PROCEDURE | void    | 0     | 0    | 0   | 26   |
| 28    | larik       | PROCEDURE | void    | 0     | 0    | 0   | 27   |
| 29    | write       | PROCEDURE | void    | 0     | 0    | 0   | 28   |
| 30    | writeln     | PROCEDURE | void    | 0     | 0    | 0   | 29   |
| 31    | read        | PROCEDURE | void    | 0     | 0    | 0   | 30   |
| 32    | readln      | PROCEDURE | void    | 0     | 0    | 0   | 31   |
| 33    | abs         | FUNCTION  | integer | 0     | 0    | 0   | 32   |
| 34    | sqr         | FUNCTION  | integer | 0     | 0    | 0   | 33   |
| 35    | sqrt        | FUNCTION  | real    | 0     | 0    | 0   | 34   |
| 36    | sin         | FUNCTION  | real    | 0     | 0    | 0   | 35   |
| 37    | cos         | FUNCTION  | real    | 0     | 0    | 0   | 36   |
| 38    | exp         | FUNCTION  | real    | 0     | 0    | 0   | 37   |
| 39    | ln          | FUNCTION  | real    | 0     | 0    | 0   | 38   |
| 40    | odd         | FUNCTION  | boolean | 0     | 0    | 0   | 39   |
| 41    | ord         | FUNCTION  | integer | 0     | 0    | 0   | 40   |
| 42    | chr         | FUNCTION  | char    | 0     | 0    | 0   | 41   |
| 43    | succ        | FUNCTION  | integer | 0     | 0    | 0   | 42   |
| 44    | pred        | FUNCTION  | integer | 0     | 0    | 0   | 43   |
| 45    | RecordBasic | PROGRAM   | void    | 0     | 0    | 0   | 44   |
| 46    | Person      | TYPE      | record  | 0     | 0    | 0   | 45   |
| 47    | p           | VARIABLE  | record  | 0     | 0    | 0   | 0    |


=====

BLOCK TABLE (btab)
=====


| Index | Last | LPar | PSize | VSize |
|-------|------|------|-------|-------|
| 0     | 47   | 0    | 0     | 1     |


[SUCCESS] Semantic analysis completed without errors.

```

Sukes menguji kemampuan semantic analyzer dalam menangani tipe data record basic, hasil uji sudah benar.

Tabel VII. FunctionBasicTest

Input																																																																																																
<pre>program FunctionBasic; variabel result: integer; fungsi Add(a, b: integer): integer; mulai Add := a + b selesai; mulai result := Add(10, 20) selesai.</pre>																																																																																																
Output																																																																																																
<pre>Decorated AST: ===== ProgramNode(name: 'FunctionBasic') └ Block → block_index:0, lev:0 └ Declarations └ VarDecl('result') → tab_index:46, type:integer, lev:0 └ FuncDecl('Add') └ Param('a','b') └ SimpleType('integer') └ Block → block_index:1, lev:1 └ CompoundStmt └ Assign → type:void, lev:1 └ Variable('Add') → tab_index:47, type:integer, lev:0 └ BinOp '+' → type:integer └ Variable('a') → tab_index:48, type:integer, lev:1 └ Variable('b') → tab_index:49, type:integer, lev:1 └ CompoundStmt └ Assign → type:void, lev:0 └ Variable('result') → tab_index:46, type:integer, lev:0 └ FuncCall('Add') └ Number(10) → type:integer └ Number(20) → type:integer ===== ===== SYMBOL TABLE (tab) ===== </pre> <table border="1"> <thead> <tr> <th>Index</th> <th>Name</th> <th>Kind</th> <th>Type</th> <th>Level</th> <th>Addr</th> <th>Ref</th> <th>Link</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>salah</td> <td>CONSTANT</td> <td>boolean</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>benar</td> <td>CONSTANT</td> <td>boolean</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>2</td> <td>integer</td> <td>TYPE</td> <td>integer</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>3</td> <td>real</td> <td>TYPE</td> <td>real</td> <td>0</td> <td>0</td> <td>0</td> <td>2</td> </tr> <tr> <td>4</td> <td>boolean</td> <td>TYPE</td> <td>boolean</td> <td>0</td> <td>0</td> <td>0</td> <td>3</td> </tr> <tr> <td>5</td> <td>char</td> <td>TYPE</td> <td>char</td> <td>0</td> <td>0</td> <td>0</td> <td>4</td> </tr> <tr> <td>6</td> <td>string</td> <td>TYPE</td> <td>string</td> <td>0</td> <td>0</td> <td>0</td> <td>5</td> </tr> <tr> <td>7</td> <td>dan</td> <td>PROCEDURE</td> <td>void</td> <td>0</td> <td>0</td> <td>0</td> <td>6</td> </tr> <tr> <td>8</td> <td>atau</td> <td>PROCEDURE</td> <td>void</td> <td>0</td> <td>0</td> <td>0</td> <td>7</td> </tr> <tr> <td>9</td> <td>tidak</td> <td>PROCEDURE</td> <td>void</td> <td>0</td> <td>0</td> <td>0</td> <td>8</td> </tr> <tr> <td>10</td> <td>bagi</td> <td>PROCEDURE</td> <td>void</td> <td>0</td> <td>0</td> <td>0</td> <td>9</td> </tr> </tbody> </table>	Index	Name	Kind	Type	Level	Addr	Ref	Link	0	salah	CONSTANT	boolean	0	0	0	0	1	benar	CONSTANT	boolean	0	0	0	0	2	integer	TYPE	integer	0	0	0	1	3	real	TYPE	real	0	0	0	2	4	boolean	TYPE	boolean	0	0	0	3	5	char	TYPE	char	0	0	0	4	6	string	TYPE	string	0	0	0	5	7	dan	PROCEDURE	void	0	0	0	6	8	atau	PROCEDURE	void	0	0	0	7	9	tidak	PROCEDURE	void	0	0	0	8	10	bagi	PROCEDURE	void	0	0	0	9
Index	Name	Kind	Type	Level	Addr	Ref	Link																																																																																									
0	salah	CONSTANT	boolean	0	0	0	0																																																																																									
1	benar	CONSTANT	boolean	0	0	0	0																																																																																									
2	integer	TYPE	integer	0	0	0	1																																																																																									
3	real	TYPE	real	0	0	0	2																																																																																									
4	boolean	TYPE	boolean	0	0	0	3																																																																																									
5	char	TYPE	char	0	0	0	4																																																																																									
6	string	TYPE	string	0	0	0	5																																																																																									
7	dan	PROCEDURE	void	0	0	0	6																																																																																									
8	atau	PROCEDURE	void	0	0	0	7																																																																																									
9	tidak	PROCEDURE	void	0	0	0	8																																																																																									
10	bagi	PROCEDURE	void	0	0	0	9																																																																																									

11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	FunctionBasic	PROGRAM	void	0	0	0	44
46	result	VARIABLE	integer	0	0	0	0
47	Add	FUNCTION	integer	0	0	1	46
48	a	VARIABLE	integer	1	3	0	0
49	b	VARIABLE	integer	1	4	0	48

=====

BLOCK TABLE (btab)

=====

Index	Last	LPar	PSize	VSize
0	47	0	0	1
1	49	0	0	2

[SUCCESS] Semantic analysis completed without errors.

Bukti dan Keterangan

```

● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test7.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'FunctionBasic')
└ Block → block_index:0, lev:0
   └ Declarations
      └ VarDecl('result') → tab_index:46, type:integer, lev:0
         └ FuncDecl('Add')
            └ Param('a', 'b')
            └ SimpleType('integer')
            └ Block → block_index:1, lev:1
               └ CompoundStmt
                  └ Assign → type:void, lev:1
                     └ Variable('Add') → tab_index:47, type:integer, lev:0
                     └ BinOp '+' → type:integer
                        └ Variable('a') → tab_index:48, type:integer, lev:1
                        └ Variable('b') → tab_index:49, type:integer, lev:1
      └ CompoundStmt
         └ Assign → type:void, lev:0
            └ Variable('result') → tab_index:46, type:integer, lev:0
            └ FuncCall('Add')
               └ Number(10) → type:integer
               └ Number(20) → type:integer
=====

=====
SYMBOL TABLE (tab)
=====

```

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	FunctionBasic	PROGRAM	void	0	0	0	44
46	result	VARIABLE	integer	0	0	0	0
47	Add	FUNCTION	integer	0	0	1	46
48	a	VARIABLE	integer	1	3	0	0
49	b	VARIABLE	integer	1	4	0	48

```
=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----+
0      47     0      0       1
1      49     0      0       2
-----+
[SUCCESS] Semantic analysis completed without errors.
```

Sukses menguji kemampuan semantic analyzer dalam menangani operasi function basic, hasil uji sudah benar.

Tabel VIII. ProcedureParamsTest

Input
<pre>program ProcedureParams; variabel x, y: integer; prosedur Swap(a, b: integer); variabel temp: integer; mulai temp := a; a := b; b := temp selesai; mulai x := 10; y := 20; Swap(x, y) selesai.</pre>
Output
<pre>Decorated AST: ===== ProgramNode(name: 'ProcedureParams') └ Block → block_index:0, lev:0 └ Declarations └ VarDecl('x','y') → tab_index:47, type:integer, lev:0 └ ProcDecl('Swap') └ Param('a','b') └ Block → block_index:1, lev:1 └ Declarations └ VarDecl('temp') → tab_index:51, type:integer, lev:1 └ CompoundStmt └ Assign → type:void, lev:1 └ Variable('temp') → tab_index:51, type:integer, lev:1 └ Variable('a') → tab_index:49, type:integer, lev:1 └ Assign → type:void, lev:1 └ Variable('a') → tab_index:49, type:integer, lev:1 └ Variable('b') → tab_index:50, type:integer, lev:1 └ Assign → type:void, lev:1 └ Variable('b') → tab_index:50, type:integer, lev:1 └ Variable('temp') → tab_index:51, type:integer, lev:1 └ CompoundStmt └ Assign → type:void, lev:0</pre>

```

    └─ Variable('x') → tab_index:46, type:integer, lev:0
        └─ Number(10) → type:integer
    └─ Assign → type:void, lev:0
        └─ Variable('y') → tab_index:47, type:integer, lev:0
            └─ Number(20) → type:integer
    └─ ProcCall('Swap') → tab_index:48, type:void, lev:0
        └─ Variable('x') → tab_index:46, type:integer, lev:0
            └─ Variable('y') → tab_index:47, type:integer, lev:0
=====
=====
```

SYMBOL TABLE (tab)

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	ProcedureParams	PROGRAM	void	0	0	0	44
46	x	VARIABLE	integer	0	0	0	0

```

47      y          VARIABLE   integer    0     1     0     46
48      Swap       PROCEDURE void      0     0     1     47
49      a          VARIABLE   integer    1     3     0     0
50      b          VARIABLE   integer    1     4     0     49
51      temp       VARIABLE  integer    1     5     0     50
=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----
0      48     0      0       2
1      51     0      0       3

```

[SUCCESS] Semantic analysis completed without errors.

Bukti dan Keterangan

```

● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test8.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'ProcedureParams')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('x','y') → tab_index:47, type:integer, lev:0
    └ ProcDecl('Swap')
    └ Param('a','b')
      └ Block → block_index:1, lev:1
        └ Declarations
          └ VarDecl('temp') → tab_index:51, type:integer, lev:1
        └ CompoundStmt
          └ Assign → type:void, lev:1
            └ Variable('temp') → tab_index:51, type:integer, lev:1
            └ Variable('a') → tab_index:49, type:integer, lev:1
          └ Assign → type:void, lev:1
            └ Variable('a') → tab_index:49, type:integer, lev:1
            └ Variable('b') → tab_index:50, type:integer, lev:1
          └ Assign → type:void, lev:1
            └ Variable('b') → tab_index:50, type:integer, lev:1
            └ Variable('temp') → tab_index:51, type:integer, lev:1
        └ CompoundStmt
          └ Assign → type:void, lev:0
            └ Variable('x') → tab_index:46, type:integer, lev:0
            └ Number(18) → type:integer
          └ Assign → type:void, lev:0
            └ Variable('y') → tab_index:47, type:integer, lev:0
            └ Number(20) → type:integer
          └ ProcCall('Swap') → tab_index:48, type:void, lev:0
            └ Variable('x') → tab_index:46, type:integer, lev:0
            └ Variable('y') → tab_index:47, type:integer, lev:0
=====
SYMBOL TABLE (tab)
=====
Index  Name   Kind   Type   Level  Addr  Ref  Link
-----
0      salah  CONSTANT boolean  0     0     0     0
1      benar  CONSTANT boolean  0     0     0     0
2      integer TYPE   integer  0     0     0     1

```

```

3   real           TYPE    real      0   0   0   2
4   boolean        TYPE    boolean   0   0   0   3
5   char           TYPE    char     0   0   0   4
6   string          TYPE   string   0   0   0   5
7   dan            PROCEDURE void    0   0   0   6
8   atau           PROCEDURE void    0   0   0   7
9   tidak          PROCEDURE void    0   0   0   8
10  bagi           PROCEDURE void    0   0   0   9
11  mod            PROCEDURE void    0   0   0  10
12  program        PROCEDURE void    0   0   0  11
13  variabel       PROCEDURE void    0   0   0  12
14  konstanta     PROCEDURE void    0   0   0  13
15  tipe           PROCEDURE void    0   0   0  14
16  prosedur       PROCEDURE void    0   0   0  15
17  fungsi         PROCEDURE void    0   0   0  16
18  mulai          PROCEDURE void    0   0   0  17
19  selesai        PROCEDURE void    0   0   0  18
20  jika           PROCEDURE void    0   0   0  19
21  maka           PROCEDURE void    0   0   0  20
22  selain-itu    PROCEDURE void    0   0   0  21
23  selama         PROCEDURE void    0   0   0  22
24  lakukan       PROCEDURE void    0   0   0  23
25  untuk          PROCEDURE void    0   0   0  24
26  ke              PROCEDURE void    0   0   0  25
27  turun-ke       PROCEDURE void    0   0   0  26
28  larik          PROCEDURE void    0   0   0  27
29  write           PROCEDURE void    0   0   0  28
30  writeln        PROCEDURE void    0   0   0  29
31  read            PROCEDURE void    0   0   0  30
32  readln         PROCEDURE void    0   0   0  31
33  abs             FUNCTION integer 0   0   0  32
34  sqr             FUNCTION integer 0   0   0  33
35  sqrt            FUNCTION real    0   0   0  34
36  sin             FUNCTION real    0   0   0  35
37  cos             FUNCTION real    0   0   0  36
38  exp             FUNCTION real    0   0   0  37
39  ln               FUNCTION real    0   0   0  38
40  odd             FUNCTION boolean 0   0   0  39
41  ord             FUNCTION integer 0   0   0  40
42  chr             FUNCTION char    0   0   0  41
43  succ            FUNCTION integer 0   0   0  42
44  pred            FUNCTION integer 0   0   0  43
45  ProcedureParams PROGRAM void    0   0   0  44
46  x                VARIABLE integer 0   0   0  45

47  y                VARIABLE integer 0   1   0   46
48  Swap            PROCEDURE void    0   0   1   47
49  a                VARIABLE integer 1   3   0   48
50  b                VARIABLE integer 1   4   0   49
51  temp            VARIABLE integer 1   5   0   50

=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----
0      48     0      0      2
1      51     0      0      3

[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani operasi prosedur dengan parameter, hasil uji sudah benar.

Tabel IX. ComplexExpressionTest

Input				
<pre> program ComplexExpressions; variabel a, b, c: integer; x, y: real; flag: boolean; mulai a := 10; b := 20; c := 30; </pre>				

```

x := (a + b) * c;
y := x / 2.0;
flag := (a < b) dan (b < c);
flag := tidak flag atau (a = 10)
selesai.

```

Output

Decorated AST:

```

=====
ProgramNode(name: 'ComplexExpressions')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('a','b','c') → tab_index:48, type:integer, lev:0
    └ VarDecl('x','y') → tab_index:50, type:real, lev:0
    └ VarDecl('flag') → tab_index:51, type:boolean, lev:0
  └ CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('a') → tab_index:46, type:integer, lev:0
      └ Number(10) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('b') → tab_index:47, type:integer, lev:0
      └ Number(20) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('c') → tab_index:48, type:integer, lev:0
      └ Number(30) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('x') → tab_index:49, type:real, lev:0
      └ BinOp '*' → type:integer
        └ BinOp '+' → type:integer
          └ Variable('a') → tab_index:46, type:integer, lev:0
          └ Variable('b') → tab_index:47, type:integer, lev:0
          └ Variable('c') → tab_index:48, type:integer, lev:0
    └ Assign → type:void, lev:0
      └ Variable('y') → tab_index:50, type:real, lev:0
      └ BinOp '/' → type:real
        └ Variable('x') → tab_index:49, type:real, lev:0
        └ Number(2.0) → type:real
    └ Assign → type:void, lev:0
      └ Variable('flag') → tab_index:51, type:boolean, lev:0
      └ BinOp 'dan' → type:boolean
        └ BinOp '<' → type:boolean
          └ Variable('a') → tab_index:46, type:integer, lev:0
          └ Variable('b') → tab_index:47, type:integer, lev:0
        └ BinOp '<' → type:boolean
          └ Variable('b') → tab_index:47, type:integer, lev:0
          └ Variable('c') → tab_index:48, type:integer, lev:0
    └ Assign → type:void, lev:0
      └ Variable('flag') → tab_index:51, type:boolean, lev:0
      └ BinOp 'atau' → type:boolean
        └ UnaryOp 'tidak'
          └ Variable('flag') → tab_index:51, type:boolean, lev:0
        └ BinOp '=' → type:boolean
          └ Variable('a') → tab_index:46, type:integer, lev:0
          └ Number(10) → type:integer
=====

=====
SYMBOL TABLE (tab)
=====

```

Index	Name	Kind	Type	Level	Addr	Ref	Link

0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	ComplexExpressions	PROGRAM	void	0	0	0	44
46	a	VARIABLE	integer	0	0	0	0
47	b	VARIABLE	integer	0	1	0	46
48	c	VARIABLE	integer	0	2	0	47
49	x	VARIABLE	real	0	3	0	48
50	y	VARIABLE	real	0	4	0	49
51	flag	VARIABLE	boolean	0	5	0	50

=====

BLOCK TABLE (btab)

=====

Index	Last	LPar	PSize	VSize
0	51	0	0	6

[SUCCESS] Semantic analysis completed without errors.

Bukti dan Keterangan

```
PS D:\Muhammad Raihan Nazim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test9.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'ComplexExpressions')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('a','b','c') → tab_index:48, type:integer, lev:0
    └ VarDecl('x','y') → tab_index:50, type:real, lev:0
    └ VarDecl('flag') → tab_index:51, type:boolean, lev:0
  └ CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('a') → tab_index:46, type:integer, lev:0
      └ Number(10) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('b') → tab_index:47, type:integer, lev:0
      └ Number(20) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('c') → tab_index:48, type:integer, lev:0
      └ Number(30) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('x') → tab_index:49, type:real, lev:0
      └ BinOp '*' → type:integer
        └ BinOp 'i' → type:integer
          └ Variable('a') → tab_index:46, type:integer, lev:0
          └ Variable('b') → tab_index:47, type:integer, lev:0
          └ Variable('c') → tab_index:48, type:integer, lev:0
    └ Assign → type:void, lev:0
      └ Variable('y') → tab_index:50, type:real, lev:0
      └ BinOp '/' → type:real
        └ Variable('x') → tab_index:49, type:real, lev:0
        └ Number(2.0) → type:real
    └ Assign → type:void, lev:0
      └ Variable('flag') → tab_index:51, type:boolean, lev:0
      └ BinOp 'dan' → type:boolean
        └ BinOp '<' → type:boolean
          └ Variable('a') → tab_index:46, type:integer, lev:0
          └ Variable('b') → tab_index:47, type:integer, lev:0
        └ BinOp '<' → type:boolean
          └ Variable('b') → tab_index:47, type:integer, lev:0
          └ Variable('c') → tab_index:48, type:integer, lev:0
    └ Assign → type:void, lev:0
      └ Variable('flag') → tab_index:51, type:boolean, lev:0
```

```

└─ BinOp 'atau' → type:boolean
   └─ UnaryOp 'tidak'
      └─ Variable('flag') → tab_index:51, type:boolean, lev:0
   └─ BinOp '=' → type:boolean
      └─ Variable('a') → tab_index:46, type:integer, lev:0
      └─ Number(10) → type:integer
=====

=====
SYMBOL TABLE (tab)
=====
Index Name Kind Type Level Addr Ref Link
0 salah CONSTANT boolean 0 0 0 0
1 benar CONSTANT boolean 0 0 0 0
2 integer TYPE integer 0 0 0 1
3 real TYPE real 0 0 0 2
4 boolean TYPE boolean 0 0 0 3
5 char TYPE char 0 0 0 4
6 string TYPE string 0 0 0 5
7 dan PROCEDURE void 0 0 0 6
8 atau PROCEDURE void 0 0 0 7
9 tidak PROCEDURE void 0 0 0 8
10 bagi PROCEDURE void 0 0 0 9
11 mod PROCEDURE void 0 0 0 10
12 program PROCEDURE void 0 0 0 11
13 variabel PROCEDURE void 0 0 0 12
14 konstanta PROCEDURE void 0 0 0 13
15 tipe PROCEDURE void 0 0 0 14
16 prosedur PROCEDURE void 0 0 0 15
17 fungsi PROCEDURE void 0 0 0 16
18 mulai PROCEDURE void 0 0 0 17
19 selesai PROCEDURE void 0 0 0 18
20 jika PROCEDURE void 0 0 0 19
21 maka PROCEDURE void 0 0 0 20
22 selain-itu PROCEDURE void 0 0 0 21
23 selama PROCEDURE void 0 0 0 22
24 lakukan PROCEDURE void 0 0 0 23
25 untuk PROCEDURE void 0 0 0 24
26 ke PROCEDURE void 0 0 0 25
27 turun-ke PROCEDURE void 0 0 0 26
28 larik PROCEDURE void 0 0 0 27
29 write PROCEDURE void 0 0 0 28
30 writeln PROCEDURE void 0 0 0 29
31 read PROCEDURE void 0 0 0 30
32 readln PROCEDURE void 0 0 0 31
33 abs FUNCTION integer 0 0 0 32
34 sqr FUNCTION integer 0 0 0 33
35 sqrt FUNCTION real 0 0 0 34
36 sin FUNCTION real 0 0 0 35
37 cos FUNCTION real 0 0 0 36
38 exp FUNCTION real 0 0 0 37
39 ln FUNCTION real 0 0 0 38
40 odd FUNCTION boolean 0 0 0 39
41 ord FUNCTION integer 0 0 0 40
42 chr FUNCTION char 0 0 0 41
43 succ FUNCTION integer 0 0 0 42
44 pred FUNCTION integer 0 0 0 43
45 ComplexExpressions PROGRAM void 0 0 0 44
46 a VARIABLE integer 0 0 0 0
47 b VARIABLE integer 0 1 0 46
48 c VARIABLE integer 0 2 0 47
49 x VARIABLE real 0 3 0 48
50 y VARIABLE real 0 4 0 49
51 flag VARIABLE boolean 0 5 0 50
=====

=====
BLOCK TABLE (btab)
=====
Index Last LPar PSize VSize
0 51 0 0 6
[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani bentuk ekspresi yang kompleks, hasil uji sudah benar.

Tabel X. ControlFlowTest

Input

```

program ControlFlow;
variabel
    x, y, max: integer;
    i, sum: integer;
mulai
    x := 10;
    y := 20;

    jika x > y maka
        max := x
    selain-itu
        max := y;

    sum := 0;
    i := 1;
    selama i <= 10 lakukan
    mulai
        sum := sum + i;
        i := i + 1
    selesai
selesai.

```

Output

Decorated AST:

```

=====
ProgramNode(name: 'ControlFlow')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('x','y','max') → tab_index:48, type:integer, lev:0
    └ VarDecl('i','sum') → tab_index:50, type:integer, lev:0
  └ CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('x') → tab_index:46, type:integer, lev:0
      └ Number(10) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('y') → tab_index:47, type:integer, lev:0
      └ Number(20) → type:integer
    └ IfStmt
      └ BinOp '>' → type:boolean
        └ Variable('x') → tab_index:46, type:integer, lev:0
        └ Variable('y') → tab_index:47, type:integer, lev:0
      └ Assign → type:void, lev:0
        └ Variable('max') → tab_index:48, type:integer, lev:0
        └ Variable('x') → tab_index:46, type:integer, lev:0
      └ Assign → type:void, lev:0
        └ Variable('max') → tab_index:48, type:integer, lev:0
        └ Variable('y') → tab_index:47, type:integer, lev:0
      └ Assign → type:void, lev:0
        └ Variable('sum') → tab_index:50, type:integer, lev:0
        └ Number(0) → type:integer
      └ Assign → type:void, lev:0
        └ Variable('i') → tab_index:49, type:integer, lev:0
        └ Number(1) → type:integer
    └ WhileStmt
      └ BinOp '<=' → type:boolean
        └ Variable('i') → tab_index:49, type:integer, lev:0
        └ Number(10) → type:integer
  └ CompoundStmt

```

```

    └─ Assign → type:void, lev:0
        └─ Variable('sum') → tab_index:50, type:integer, lev:0
            └─ BinOp '+' → type:integer
                └─ Variable('sum') → tab_index:50, type:integer, lev:0
                    └─ Variable('i') → tab_index:49, type:integer, lev:0
    └─ Assign → type:void, lev:0
        └─ Variable('i') → tab_index:49, type:integer, lev:0
            └─ BinOp '+' → type:integer
                └─ Variable('i') → tab_index:49, type:integer, lev:0
                    └─ Number(1) → type:integer
=====
```

=====
=====
SYMBOL TABLE (tab)
=====

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43

```

45 ControlFlow      PROGRAM    void      0      0      0      44
46 x                VARIABLE   integer   0      0      0      0
47 y                VARIABLE   integer   0      1      0      46
48 max              VARIABLE   integer   0      2      0      47
49 i                VARIABLE   integer   0      3      0      48
50 sum              VARIABLE   integer   0      4      0      49

=====
BLOCK TABLE (btab)
=====
Index  Last   LPar   PSize   VSize
-----
0      50     0      0       5

[SUCCESS] Semantic analysis completed without errors.

```

Bukti dan Keterangan

```

● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test10.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'ControlFlow')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('x', 'y', 'max') → tab_index:48, type:integer, lev:0
    └ VarDecl('i', 'sum') → tab_index:50, type:integer, lev:0
  └ CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('x') → tab_index:46, type:integer, lev:0
      └ Number(10) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('y') → tab_index:47, type:integer, lev:0
      └ Number(20) → type:integer
    └ Ifstmt
      └ BinOp '>' → type:boolean
        └ Variable('x') → tab_index:46, type:integer, lev:0
        └ Variable('y') → tab_index:47, type:integer, lev:0
      └ Assign → type:void, lev:0
        └ Variable('max') → tab_index:48, type:integer, lev:0
        └ Variable('x') → tab_index:46, type:integer, lev:0
      └ Assign → type:void, lev:0
        └ Variable('max') → tab_index:48, type:integer, lev:0
        └ Variable('y') → tab_index:47, type:integer, lev:0
    └ Assign → type:void, lev:0
      └ Variable('sum') → tab_index:50, type:integer, lev:0
      └ Number(0) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('i') → tab_index:49, type:integer, lev:0
      └ Number(1) → type:integer
  └ Whilestmt
    └ BinOp '<=' → type:boolean
      └ Variable('i') → tab_index:49, type:integer, lev:0
      └ Number(10) → type:integer
    └ CompoundStmt
      └ Assign → type:void, lev:0
        └ Variable('sum') → tab_index:50, type:integer, lev:0
      └ BinOp '+' → type:integer
        └ Variable('sum') → tab_index:50, type:integer, lev:0
        └ Variable('i') → tab_index:49, type:integer, lev:0
      └ Assign → type:void, lev:0

```

```

    └─ Variable('i') → tab_index:49, type:integer, lev:0
      BinOp '+' → type:integer
        └─ Variable('i') → tab_index:49, type:integer, lev:0
          Number(1) → type:integer
=====

=====

SYMBOL TABLE (tab)

-----  


| Index | Name       | Kind      | Type    | Level | Addr | Ref | Link |
|-------|------------|-----------|---------|-------|------|-----|------|
| 0     | salah      | CONSTANT  | boolean | 0     | 0    | 0   | 0    |
| 1     | benar      | CONSTANT  | boolean | 0     | 0    | 0   | 0    |
| 2     | integer    | TYPE      | integer | 0     | 0    | 0   | 1    |
| 3     | real       | TYPE      | real    | 0     | 0    | 0   | 2    |
| 4     | boolean    | TYPE      | boolean | 0     | 0    | 0   | 3    |
| 5     | char       | TYPE      | char    | 0     | 0    | 0   | 4    |
| 6     | string     | TYPE      | string  | 0     | 0    | 0   | 5    |
| 7     | dan        | PROCEDURE | void    | 0     | 0    | 0   | 6    |
| 8     | atau       | PROCEDURE | void    | 0     | 0    | 0   | 7    |
| 9     | tidak      | PROCEDURE | void    | 0     | 0    | 0   | 8    |
| 10    | bagi       | PROCEDURE | void    | 0     | 0    | 0   | 9    |
| 11    | mod        | PROCEDURE | void    | 0     | 0    | 0   | 10   |
| 12    | program    | PROCEDURE | void    | 0     | 0    | 0   | 11   |
| 13    | variabel   | PROCEDURE | void    | 0     | 0    | 0   | 12   |
| 14    | konstanta  | PROCEDURE | void    | 0     | 0    | 0   | 13   |
| 15    | tipe       | PROCEDURE | void    | 0     | 0    | 0   | 14   |
| 16    | prosedur   | PROCEDURE | void    | 0     | 0    | 0   | 15   |
| 17    | fungsi     | PROCEDURE | void    | 0     | 0    | 0   | 16   |
| 18    | mulai      | PROCEDURE | void    | 0     | 0    | 0   | 17   |
| 19    | selesai    | PROCEDURE | void    | 0     | 0    | 0   | 18   |
| 20    | jika       | PROCEDURE | void    | 0     | 0    | 0   | 19   |
| 21    | maka       | PROCEDURE | void    | 0     | 0    | 0   | 20   |
| 22    | selain-itu | PROCEDURE | void    | 0     | 0    | 0   | 21   |
| 23    | selama     | PROCEDURE | void    | 0     | 0    | 0   | 22   |
| 24    | lakukan    | PROCEDURE | void    | 0     | 0    | 0   | 23   |
| 25    | untuk      | PROCEDURE | void    | 0     | 0    | 0   | 24   |
| 26    | ke         | PROCEDURE | void    | 0     | 0    | 0   | 25   |
| 27    | turun-ke   | PROCEDURE | void    | 0     | 0    | 0   | 26   |
| 28    | larik      | PROCEDURE | void    | 0     | 0    | 0   | 27   |
| 29    | write      | PROCEDURE | void    | 0     | 0    | 0   | 28   |
| 30    | writeln    | PROCEDURE | void    | 0     | 0    | 0   | 29   |
| 31    | read       | PROCEDURE | void    | 0     | 0    | 0   | 30   |
| 32    | readln     | PROCEDURE | void    | 0     | 0    | 0   | 31   |


=====

-----  


| Index | Name        | Kind     | Type    | Level | Addr | Ref | Link |
|-------|-------------|----------|---------|-------|------|-----|------|
| 33    | abs         | FUNCTION | integer | 0     | 0    | 0   | 32   |
| 34    | sqr         | FUNCTION | integer | 0     | 0    | 0   | 33   |
| 35    | sqrt        | FUNCTION | real    | 0     | 0    | 0   | 34   |
| 36    | sin         | FUNCTION | real    | 0     | 0    | 0   | 35   |
| 37    | cos         | FUNCTION | real    | 0     | 0    | 0   | 36   |
| 38    | exp         | FUNCTION | real    | 0     | 0    | 0   | 37   |
| 39    | ln          | FUNCTION | real    | 0     | 0    | 0   | 38   |
| 40    | odd         | FUNCTION | boolean | 0     | 0    | 0   | 39   |
| 41    | ord         | FUNCTION | integer | 0     | 0    | 0   | 40   |
| 42    | chr         | FUNCTION | char    | 0     | 0    | 0   | 41   |
| 43    | succ        | FUNCTION | integer | 0     | 0    | 0   | 42   |
| 44    | pred        | FUNCTION | integer | 0     | 0    | 0   | 43   |
| 45    | ControlFlow | PROGRAM  | void    | 0     | 0    | 0   | 44   |
| 46    | x           | VARIABLE | integer | 0     | 0    | 0   | 0    |
| 47    | y           | VARIABLE | integer | 0     | 1    | 0   | 46   |
| 48    | max         | VARIABLE | integer | 0     | 2    | 0   | 47   |
| 49    | i           | VARIABLE | integer | 0     | 3    | 0   | 48   |
| 50    | sum         | VARIABLE | integer | 0     | 4    | 0   | 49   |


=====

BLOCK TABLE (btab)

-----  


| Index | Last | LPar | PSize | VSize |
|-------|------|------|-------|-------|
| 0     | 50   | 0    | 0     | 5     |


=====

[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani control flow program, hasil uji sudah benar.

Tabel XI. ErrorUndeclaredTest

Input
program ErrorUndeclared; variabel

```

x: integer;
mulai
    x := 10;
    y := 20
selesai.

```

Output

Semantic Error: Undeclared identifier 'y' (identifier: 'y')

Bukti dan Keterangan

```

PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/
test11.pas --decorated
Semantic Error: Undeclared identifier 'y' (identifier: 'y')

```

Sukses menguji kemampuan semantic analyzer dalam menangani bentuk kesalahan semantic berupa variabel yang belum dideklarasikan, hasil uji sudah benar.

Tabel XII. ErrorTypeMismatchTest

Input

```

program ErrorTypeMismatch;
variabel
    x: integer;
    flag: boolean;
mulai
    x := 10;
    flag := x
selesai.

```

Output

Semantic Error: Type mismatch: expected boolean, got integer in assignment

Bukti dan Keterangan

```

PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/
test12.pas --decorated
Semantic Error: Type mismatch: expected boolean, got integer in assignment

```

Sukses menguji kemampuan semantic analyzer dalam menangani bentuk kesalahan semantic berupa operasi assignment yang tidak sesuai antara tipe data variabel dengan valuenya, hasil uji sudah benar.

Tabel XIII. ErrorRedeclarationTest

Input

```

program ErrorRedeclaration;
variabel
    x: integer;
    x: real;
mulai
    x := 10
selesai.

```

Output

```
Semantic Error: Duplicate declaration of 'x' (identifier: 'x')
```

Bukti dan Keterangan

```
PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test13.pas --decorated
Semantic Error: Duplicate declaration of 'x' (identifier: 'x')
```

Sukses menguji kemampuan semantic analyzer dalam menangani bentuk kesalahan semantic berupa variabel yang dilakukan deklarasi lebih dari sekali, hasil uji sudah benar.

Tabel XIV. MultidimArrayTest

Input

```
program MultidimArray;
variabel
    matrix: larik[1..3, 1..3] dari integer;
    i, j: integer;
mulai
    i := 1;
    j := 1;
    matrix[i, j] := 100;
    matrix[2, 3] := 200
selesai.
```

Output

Decorated AST:

```
=====
ProgramNode(name: 'MultidimArray')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('matrix') → tab_index:46, type:array of array of integer, lev:0
    └ VarDecl('i','j') → tab_index:48, type:integer, lev:0
  └ CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('i') → tab_index:47, type:integer, lev:0
      └ Number(1) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('j') → tab_index:48, type:integer, lev:0
      └ Number(1) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('matrix[]') → tab_index:46, type:integer, lev:0
        └ Variable('i') → tab_index:47, type:integer, lev:0
        └ Variable('j') → tab_index:48, type:integer, lev:0
        └ Number(100) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('matrix[]') → tab_index:46, type:integer, lev:0
        └ Number(2) → type:integer
        └ Number(3) → type:integer
        └ Number(200) → type:integer
=====
```

```
=====
SYMBOL TABLE (tab)
=====
```

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	MultidimArray	PROGRAM	void	0	0	0	44
46	matrix	VARIABLE	array of array of integer	0	0	0	1
0							
47	i	VARIABLE	integer	0	1	0	46
48	j	VARIABLE	integer	0	2	0	47
=====							
ARRAY TABLE (atab)							
=====							
Index	IdxType	ElemType	Low	High	ElemSz	Size	
0	integer	integer	1	3	1	3	
1	integer	array of integer	1	3	3	9	
2							
=====							

```
BLOCK TABLE (btab)
=====
Index  Last      LPar      PSize     VSize
-----
0       48        0         0         3

[SUCCESS] Semantic analysis completed without errors.
```

Bukti dan Keterangan

```
PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test14.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'MultidimArray')
└ Block → block_index:0, lev:0
  └ Declarations
    └ VarDecl('matrix') → tab_index:46, type:array of array of integer, lev:0
    └ VarDecl('i', 'j') → tab_index:48, type:integer, lev:0
  └ CompoundStmt
    └ Assign → type:void, lev:0
      └ Variable('i') → tab_index:47, type:integer, lev:0
      └ Number(1) → type:integer
    └ Assign → type:void, lev:0
      └ Variable('j') → tab_index:48, type:integer, lev:0
      └ Number(1) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('matrix[]') → tab_index:46, type:integer, lev:0
      └ Variable('i') → tab_index:47, type:integer, lev:0
      └ Variable('j') → tab_index:48, type:integer, lev:0
      └ Number(100) → type:integer
    └ Assign → type:void, lev:0
      └ VarAccess('matrix[]') → tab_index:46, type:integer, lev:0
      └ Number(2) → type:integer
      └ Number(3) → type:integer
      └ Number(200) → type:integer
=====

=====
SYMBOL TABLE (tab)
=====
Index  Name          Kind      Type      Level   Addr   Ref   Link
-----
0      salah        CONSTANT  boolean   0       0     0     0
1      benar        CONSTANT  boolean   0       0     0     0
2      integer      TYPE     integer   0       0     0     1
3      real         TYPE     real     0       0     0     2
4      boolean      TYPE     boolean  0       0     0     3
5      char         TYPE     char    0       0     0     4
6      string       TYPE     string  0       0     0     5
7      dan          PROCEDURE void    0       0     0     6
8      atau         PROCEDURE void    0       0     0     7
9      tidak        PROCEDURE void    0       0     0     8
```

```

10  bagi      PROCEDURE void    0   0   0   9
11  mod       PROCEDURE void    0   0   0   10
12  program   PROCEDURE void    0   0   0   11
13  variabel  PROCEDURE void    0   0   0   12
14  konstanta PROCEDURE void    0   0   0   13
15  tipe      PROCEDURE void    0   0   0   14
16  prosedur  PROCEDURE void    0   0   0   15
17  fungsi   PROCEDURE void    0   0   0   16
18  mulai    PROCEDURE void    0   0   0   17
19  selesai  PROCEDURE void    0   0   0   18
20  jika     PROCEDURE void    0   0   0   19
21  maka     PROCEDURE void    0   0   0   20
22  selain-itu PROCEDURE void    0   0   0   21
23  selama   PROCEDURE void    0   0   0   22
24  lakukan  PROCEDURE void    0   0   0   23
25  untuk    PROCEDURE void    0   0   0   24
26  ke       PROCEDURE void    0   0   0   25
27  turun-ke PROCEDURE void    0   0   0   26
28  larik    PROCEDURE void    0   0   0   27
29  write    PROCEDURE void    0   0   0   28
30  writeln  PROCEDURE void    0   0   0   29
31  read    PROCEDURE void    0   0   0   30
32  readln  PROCEDURE void    0   0   0   31
33  abs     FUNCTION integer  0   0   0   32
34  sqr     FUNCTION integer  0   0   0   33
35  sqrt    FUNCTION real    0   0   0   34
36  sin     FUNCTION real    0   0   0   35
37  cos     FUNCTION real    0   0   0   36
38  exp     FUNCTION real    0   0   0   37
39  ln      FUNCTION real    0   0   0   38
40  odd    FUNCTION boolean  0   0   0   39
41  ord     FUNCTION integer 0   0   0   40
42  chr     FUNCTION char    0   0   0   41
43  succ    FUNCTION integer 0   0   0   42
44  pred    FUNCTION integer 0   0   0   43
45  MultidimArray PROGRAM void    0   0   0   44
46  matrix  VARIABLE array of array of integer 0   0   1   0
47  i       VARIABLE integer  0   1   0   46
48  j       VARIABLE integer  0   2   0   47

```

```

=====
ARRAY TABLE (atab)
=====
Index  IdxType   ElemType   Low   High   ElemSz   Size
-----
0      integer    integer    1     3      1       3
1      integer    array of integer 1   3      3       9
=====

=====
BLOCK TABLE (btab)
=====
Index  Last    LPar    PSize   VSize
-----
0      48     0      0       3
=====

[SUCCESS] Semantic analysis completed without errors.

```

Sukses menguji kemampuan semantic analyzer dalam menangani tipe data multidimensional array (matrix), hasil uji sudah benar.

Tabel XV. ComprehensiveTest

Input
<pre> program Comprehensive; tipe Student = rekaman id: integer; name: larik[1..50] dari char; gpa: real; selesai; variabel students: larik[1..100] dari Student; count: integer; avg_gpa: real; </pre>

```

prosedur InitStudent(s: Student; student_id: integer);
mulai
    s.id := student_id;
    s.gpa := 0.0
selesai;

fungsi CalculateAverage(n: integer): real;
variabel
    i: integer;
    total: real;
mulai
    total := 0.0;
    i := 1;
    selama i <= n lakukan
    mulai
        total := total + students[i].gpa;
        i := i + 1
    selesai;
    CalculateAverage := total / n
selesai;

mulai
    count := 3;
    InitStudent(students[1], 1001);
    students[1].gpa := 3.5;
    InitStudent(students[2], 1002);
    students[2].gpa := 3.8;
    InitStudent(students[3], 1003);
    students[3].gpa := 3.2;
    avg_gpa := CalculateAverage(count)
selesai.

```

Output

Decorated AST:

```

=====
ProgramNode(name: 'Comprehensive')
└ Block → block_index:0, lev:0
    └ Declarations
        └ TypeDecl('Student')
            └ RecordType
                └ VarDecl('id')
                └ VarDecl('name')
                └ VarDecl('gpa')
        └ VarDecl('students') → tab_index:47, type:array of record, lev:0
        └ VarDecl('count') → tab_index:48, type:integer, lev:0
        └ VarDecl('avg_gpa') → tab_index:49, type:real, lev:0
    └ ProcDecl('InitStudent')
        └ Param('s')
        └ Param('student_id')
            └ Block → block_index:1, lev:1
                └ CompoundStmt
                    └ Assign → type:void, lev:1
                        └ VarAccess('.id') → tab_index:51, type:integer, lev:1
                            └ Variable('student_id') → tab_index:52, type:integer, lev:1
                        └ Assign → type:void, lev:1
                            └ VarAccess('.gpa') → tab_index:51, type:real, lev:1
                                └ Number(0.0) → type:real
        └ FuncDecl('CalculateAverage')
            └ Param('n')
            └ SimpleType('real')

```

```

└ Block → block_index:2, lev:1
  └ Declarations
    └ VarDecl('i') → tab_index:55, type:integer, lev:1
    └ VarDecl('total') → tab_index:56, type:real, lev:1
  └ CompoundStmt
    └ Assign → type:void, lev:1
      └ Variable('total') → tab_index:56, type:real, lev:1
      └ Number(0.0) → type:real
    └ Assign → type:void, lev:1
      └ Variable('i') → tab_index:55, type:integer, lev:1
      └ Number(1) → type:integer
    └ WhileStmt
      └ BinOp '<=' → type:boolean
        └ Variable('i') → tab_index:55, type:integer, lev:1
        └ Variable('n') → tab_index:54, type:integer, lev:1
      └ CompoundStmt
        └ Assign → type:void, lev:1
          └ Variable('total') → tab_index:56, type:real, lev:1
          └ BinOp '+' → type:real
            └ Variable('total') → tab_index:56, type:real, lev:1
            └ VarAccess('students[]') → tab_index:47, type:record, lev:0
              └ Variable('i') → tab_index:55, type:integer, lev:1
              └ VarAccess('.gpa') → type:real
        └ Assign → type:void, lev:1
          └ Variable('i') → tab_index:55, type:integer, lev:1
          └ BinOp '+' → type:integer
            └ Variable('i') → tab_index:55, type:integer, lev:1
            └ Number(1) → type:integer
        └ Assign → type:void, lev:1
          └ Variable('CalculateAverage') → tab_index:53, type:real, lev:0
        └ BinOp '/' → type:real
          └ Variable('total') → tab_index:56, type:real, lev:1
          └ Variable('n') → tab_index:54, type:integer, lev:1
    └ CompoundStmt
      └ Assign → type:void, lev:0
        └ Variable('count') → tab_index:48, type:integer, lev:0
        └ Number(3) → type:integer
      └ ProcCall('InitStudent') → tab_index:50, type:void, lev:0
        └ VarAccess('students[]') → tab_index:47, type:record, lev:0
          └ Number(1) → type:integer
          └ Number(1001) → type:integer
      └ Assign → type:void, lev:0
        └ VarAccess('students[]') → tab_index:47, type:record, lev:0
          └ Number(1) → type:integer
          └ VarAccess('.gpa') → type:real
          └ Number(3.5) → type:real
      └ ProcCall('InitStudent') → tab_index:50, type:void, lev:0
        └ VarAccess('students[]') → tab_index:47, type:record, lev:0
          └ Number(2) → type:integer
          └ Number(1002) → type:integer
      └ Assign → type:void, lev:0
        └ VarAccess('students[]') → tab_index:47, type:record, lev:0
          └ Number(2) → type:integer
          └ VarAccess('.gpa') → type:real
          └ Number(3.8) → type:real
      └ ProcCall('InitStudent') → tab_index:50, type:void, lev:0
        └ VarAccess('students[]') → tab_index:47, type:record, lev:0
          └ Number(3) → type:integer
          └ Number(1003) → type:integer
      └ Assign → type:void, lev:0
        └ VarAccess('students[]') → tab_index:47, type:record, lev:0
          └ Number(3) → type:integer
          └ VarAccess('.gpa') → type:real
          └ Number(3.2) → type:real
      └ Assign → type:void, lev:0
        └ Variable('avg_gpa') → tab_index:49, type:real, lev:0
        └ FuncCall('CalculateAverage')
          └ Variable('count') → tab_index:48, type:integer, lev:0
=====
```

=====							
SYMBOL TABLE (tab)							
Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	Comprehensive	PROGRAM	void	0	0	0	44
46	Student	TYPE	record	0	0	0	45
47	students	VARIABLE	array of record	0	0	1	0
48	count	VARIABLE	integer	0	1	0	47
49	avg_gpa	VARIABLE	real	0	2	0	48
50	InitStudent	PROCEDURE	void	0	0	1	49
51	s	VARIABLE	record	1	3	0	0
52	student_id	VARIABLE	integer	1	4	0	51
53	CalculateAverage	FUNCTION	real	0	0	2	50
54	n	VARIABLE	integer	1	3	0	0
55	i	VARIABLE	integer	1	4	0	54
56	total	VARIABLE	real	1	5	0	55

=====							
ARRAY TABLE (atab)							
Index	IdxType	ElemType	Low	High	ElemSz	Size	
0	integer	char	1	50	1	50	
1	integer	record	1	100	52	5200	

```
=====
BLOCK TABLE (btab)
=====
Index  Last      LPar      PSize     VSize
-----
0      53       0         0         3
1      52       0         0         2
2      56       0         0         3

[SUCCESS] Semantic analysis completed without errors.
```

Bukti dan Keterangan

```
● PS D:\Muhammad Raihan Nazhim Oktana\Kuliah STEI-K (IF-G) ITB\Tubes TBFO\PIA-Tubes-IF2224> uv run src/semantic_main.py test/milestone-3/input/test15.pas --decorated

Decorated AST:
=====
ProgramNode(name: 'Comprehensive')
└ Block → block_index:0, lev:0
   └ Declarations
      └ TypeDecl('Student')
         └ RecordType
            └ VarDecl('id')
            └ VarDecl('name')
            └ VarDecl('gpa')
   └ VarDecl('students') → tab_index:47, type:array of record, lev:0
   └ VarDecl('count') → tab_index:48, type:integer, lev:0
   └ VarDecl('avg_gpa') → tab_index:49, type:real, lev:0
   └ ProcDecl('InitStudent')
      └ Param('s')
      └ Param('student_id')
      └ Block → block_index:1, lev:1
         └ CompoundStmt
            └ Assign → type:void, lev:1
               └ VarAccess('.id') → tab_index:51, type:integer, lev:1
               └ Variable('student_id') → tab_index:52, type:integer, lev:1
            └ Assign → type:void, lev:1
               └ VarAccess('.gpa') → tab_index:51, type:real, lev:1
               └ Number(0.0) → type:real
   └ FuncDecl('CalculateAverage')
      └ Param('n')
      └ SimpleType('real')
      └ Block → block_index:2, lev:1
         └ Declarations
            └ VarDecl('i') → tab_index:55, type:integer, lev:1
            └ VarDecl('total') → tab_index:56, type:real, lev:1
         └ CompoundStmt
            └ Assign → type:void, lev:1
               └ Variable('total') → tab_index:56, type:real, lev:1
               └ Number(0.0) → type:real
            └ Assign → type:void, lev:1
               └ Variable('i') → tab_index:55, type:integer, lev:1
               └ Number(1) → type:integer
            └ WhileStmt
               └ BinOp '<=' → type:boolean
                  └ Variable('i') → tab_index:55, type:integer, lev:1
```

```

└─ Variable('n') → tab_index:54, type:integer, lev:1
CompoundStmt
└─ Assign → type:void, lev:1
    └─ Variable('total') → tab_index:56, type:real, lev:1
        └─ BinOp '+' → type:real
            └─ Variable('total') → tab_index:56, type:real, lev:1
            └─ VarAccess('students[]') → tab_index:47, type:record, lev:0
                └─ Variable('i') → tab_index:55, type:integer, lev:1
                    └─ VarAccess('.gpa') → type:real
Assign → type:void, lev:1
└─ Variable('i') → tab_index:55, type:integer, lev:1
└─ BinOp '+' → type:integer
    └─ Variable('i') → tab_index:55, type:integer, lev:1
    └─ Number(1) → type:integer
Assign → type:void, lev:1
└─ Variable('CalculateAverage') → tab_index:53, type:real, lev:0
└─ BinOp '/' → type:real
    └─ Variable('total') → tab_index:56, type:real, lev:1
    └─ Variable('n') → tab_index:54, type:integer, lev:1
CompoundStmt
└─ Assign → type:void, lev:0
    └─ Variable('count') → tab_index:48, type:integer, lev:0
    └─ Number(3) → type:integer
└─ ProcCall('InitStudent') → tab_index:50, type:void, lev:0
    └─ VarAccess('students[]') → tab_index:47, type:record, lev:0
        └─ Number(1) → type:integer
        └─ Number(1001) → type:integer
Assign → type:void, lev:0
└─ VarAccess('students[]') → tab_index:47, type:record, lev:0
    └─ Number(1) → type:integer
    └─ VarAccess('.gpa') → type:real
    └─ Number(3.5) → type:real
ProcCall('InitStudent') → tab_index:50, type:void, lev:0
└─ VarAccess('students[]') → tab_index:47, type:record, lev:0
    └─ Number(2) → type:integer
    └─ Number(1002) → type:integer
Assign → type:void, lev:0
└─ VarAccess('students[]') → tab_index:47, type:record, lev:0
    └─ Number(2) → type:integer
    └─ VarAccess('.gpa') → type:real
    └─ Number(3.8) → type:real
ProcCall('InitStudent') → tab_index:50, type:void, lev:0
└─ VarAccess('students[]') → tab_index:47, type:record, lev:0
    └─ Number(3) → type:integer

```

```

└ Number(1003) → type:integer
Assign → type:void, lev:0
  |- VarAccess('students')[] → tab_index:47, type:record, lev:0
    |- Number(3) → type:integer
      |- VarAccess('.gpa') → type:real
        Number(3.2) → type:real
Assign → type:void, lev:0
  |- Variable('avg_gpa') → tab_index:49, type:real, lev:0
  |- FuncCall('CalculateAverage')
    |- Variable('count') → tab_index:48, type:integer, lev:0
=====

=====
SYMBOL TABLE (tab)
=====

```

Index	Name	Kind	Type	Level	Addr	Ref	Link
0	salah	CONSTANT	boolean	0	0	0	0
1	benar	CONSTANT	boolean	0	0	0	0
2	integer	TYPE	integer	0	0	0	1
3	real	TYPE	real	0	0	0	2
4	boolean	TYPE	boolean	0	0	0	3
5	char	TYPE	char	0	0	0	4
6	string	TYPE	string	0	0	0	5
7	dan	PROCEDURE	void	0	0	0	6
8	atau	PROCEDURE	void	0	0	0	7
9	tidak	PROCEDURE	void	0	0	0	8
10	bagi	PROCEDURE	void	0	0	0	9
11	mod	PROCEDURE	void	0	0	0	10
12	program	PROCEDURE	void	0	0	0	11
13	variabel	PROCEDURE	void	0	0	0	12
14	konstanta	PROCEDURE	void	0	0	0	13
15	tipe	PROCEDURE	void	0	0	0	14
16	prosedur	PROCEDURE	void	0	0	0	15
17	fungsi	PROCEDURE	void	0	0	0	16
18	mulai	PROCEDURE	void	0	0	0	17
19	selesai	PROCEDURE	void	0	0	0	18
20	jika	PROCEDURE	void	0	0	0	19
21	maka	PROCEDURE	void	0	0	0	20
22	selain-itu	PROCEDURE	void	0	0	0	21
23	selama	PROCEDURE	void	0	0	0	22
24	lakukan	PROCEDURE	void	0	0	0	23
25	untuk	PROCEDURE	void	0	0	0	24
26	ke	PROCEDURE	void	0	0	0	25
27	turun-ke	PROCEDURE	void	0	0	0	26
28	larik	PROCEDURE	void	0	0	0	27
29	write	PROCEDURE	void	0	0	0	28
30	writeln	PROCEDURE	void	0	0	0	29
31	read	PROCEDURE	void	0	0	0	30
32	readln	PROCEDURE	void	0	0	0	31
33	abs	FUNCTION	integer	0	0	0	32
34	sqr	FUNCTION	integer	0	0	0	33
35	sqrt	FUNCTION	real	0	0	0	34
36	sin	FUNCTION	real	0	0	0	35
37	cos	FUNCTION	real	0	0	0	36
38	exp	FUNCTION	real	0	0	0	37
39	ln	FUNCTION	real	0	0	0	38
40	odd	FUNCTION	boolean	0	0	0	39
41	ord	FUNCTION	integer	0	0	0	40
42	chr	FUNCTION	char	0	0	0	41
43	succ	FUNCTION	integer	0	0	0	42
44	pred	FUNCTION	integer	0	0	0	43
45	Comprehensive	PROGRAM	void	0	0	0	44
46	Student	TYPE	record	0	0	0	45
47	Students	VARIABLE	array of record	0	0	1	0
48	count	VARIABLE	integer	0	1	0	47
49	avg_gpa	VARIABLE	real	0	2	0	48
50	InitStudent	PROCEDURE	void	0	0	1	49
51	s	VARIABLE	record	1	3	0	0
52	student_id	VARIABLE	integer	1	4	0	51
53	CalculateAverage	FUNCTION	real	0	0	2	50
54	n	VARIABLE	integer	1	3	0	0
55	i	VARIABLE	integer	1	4	0	54
56	total	VARIABLE	real	1	5	0	55

```

=====
ARRAY TABLE (atab)
=====
Index  IdxType   ElemType   Low    High   Elemsz  Size
-----+-----+-----+-----+-----+-----+-----+
0     integer   char       1      50     1      50
1     integer   record     1      100    52     5200
-----+-----+-----+-----+-----+-----+-----+
BLOCK TABLE (btab)
=====
Index  Last    LPar    PSize   Vsize
-----+-----+-----+-----+-----+
0     53     0      0      3
1     52     0      0      2
2     56     0      0      3
-----+-----+-----+-----+-----+

```

[SUCCESS] Semantic analysis completed without errors.

Sukses menguji kemampuan semantic analyzer dalam menangani program yang komprehensif, hasil uji sudah benar.

BAB V: Kesimpulan dan Saran

5.1 Kesimpulan

Tugas Besar Teori Bahasa Formal dan Otomata (IF2224-24) Milestone 3 ini berhasil mengimplementasikan tahap ketiga dari proses kompilasi, yaitu semantic analysis untuk bahasa Pascal-S yang telah dimodifikasi. Pada tahap ini, sistem tidak hanya mampu membangun Abstract Syntax Tree (AST) dari hasil parsing, tetapi juga melengkapi pohon sintaks tersebut dengan anotasi semantik yang akurat—meliputi tipe ekspresi, referensi simbol, serta informasi skope yang sesuai dengan prinsip static scoping.

Implementasi semantic visitor memungkinkan traversal terstruktur pada AST tanpa memerlukan modifikasi arsitektur internal node, sehingga logika pemeriksaan tipe, validasi deklarasi, pengecekan skope, serta penanganan parameter prosedur dapat dilakukan secara modular dan sistematis. Selain itu, perancangan symbol table yang terdiri dari tab, btab, dan atab sesuai model Pascal-S memastikan bahwa seluruh entitas program, mulai dari variabel, konstanta, hingga tipe komposit, dapat direpresentasikan dan diakses secara deterministik.

Dengan keberhasilan tahap ini, sistem kompilator telah mencapai kemampuan untuk memverifikasi program Pascal-S secara semantik, mendeteksi kesalahan seperti type mismatch, deklarasi ganda, hingga penggunaan identifier yang belum dideklarasikan. Milestone 3 menjadi tonggak penting karena menutup seluruh fase analisis dalam pipeline kompilasi—mulai dari lexical analysis, syntactic analysis, hingga semantic analysis—serta menyediakan fondasi yang solid bagi tahap berikutnya, yaitu intermediate code generation dan optimisasi.

5.2 Saran

Untuk pengembangan di milestone berikutnya, disarankan agar semantic analyzer diperluas dengan cakupan pemeriksaan yang lebih komprehensif, terutama untuk struktur kompleks seperti array multidimensi, record bersarang, dan parameter prosedur dengan mekanisme

by-reference. Selain itu, integrasi antara decorated AST dan generator kode juga perlu diperdalam agar informasi semantik yang telah dihimpun dapat dimanfaatkan secara optimal dalam proses produksi kode.

Ditinjau dari aspek keandalan, pengujian tambahan dengan kasus-kasus ekstrem seperti blok deeply nested, ekspresi campuran yang kompleks, dan pemanggilan prosedur rekursif sangat direkomendasikan untuk memastikan bahwa mekanisme scoping dan type checking bekerja konsisten pada seluruh kondisi. Terakhir, peningkatan pada sistem pelaporan kesalahan—misalnya dengan menyediakan konteks lokasi kesalahan dan penjelasan yang lebih ramah pengguna—akan sangat membantu proses debugging dan memperkaya pengalaman penggunaan kompilator.

Lampiran

Pranala Repositori Github: <https://github.com/fathurwithyou/PIA-Tubes-IF2224>

Pranala Workspace Diagram:

https://drive.google.com/file/d/1czvTxZ2tvb7p6O_OSo0QtYlc81k2ghwc/view

Tabel Pembagian Tugas:

NIM	Nama Lengkap	Persentase Pembagian Tugas
13523021	Muhammad Raihan Nazhim Oktana	100%
13523057	Faqih Muhammad Syuhada	100%
13523097	Shanice Feodora Tjahjono	100%
13523105	Muhammad Fathur Rizky	100%

Daftar Pustaka

- [1] A. Aho, M. Lam, R. Sethi, and J. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd ed. Boston, MA: Addison-Wesley, 2007.
- [2] D. E. Knuth, “Semantics of Context-Free Languages,” *Mathematical Systems Theory*, vol. 2, no. 2, pp. 127–145, 1968.
- [3] N. Wirth, *Pascal-S: A Subset and Its Implementation*. Zürich: Institut für Informatik ETH, 1974. [Online].
- [4] R. Spivak, “Let’s Build A Simple Interpreter (Part 7): Abstract Syntax Trees,” 2015. [Online]. Available: <https://ruslanspivak.com/lbasi-part7/>
- [5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley, 1994.

[6] University of Liège, “Semantic Analysis — Compilation Course Notes,” 2015. [Online]. Available:

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

[7] H. H. Otten, “Pascal for Small Machines,” 2010. [Online]. Available: <http://pascal.hansotten.com/niklaus-wirth/pascal-s/>

[8] TextX Project, “Example of Parse Trees using Arpeggio Parser Generator,” 2020. [Online]. Available: https://textx.github.io/Arpeggio/latest/images/calc_parse_tree.dot.png