

Báo cáo LAB02

IT3323 Mã lớp 161269

Bài 4 : Quản lý phạm vi

Họ và tên: Dương Công Thuyết

MSSV: 20225932

I. Nội dung chính về quản lí phạm vi

1. Khái niệm Scope Management

- Scope (phạm vi) là vùng trong chương trình mà một định danh (identifier) có hiệu lực và có thể được truy cập.
- Scope Management là cơ chế quản lý việc khai báo, tìm kiếm và kiểm tra tính hợp lệ của các định danh trong các phạm vi khác nhau của chương trình.

2. Cấu trúc dữ liệu chính

a) Scope (symtab.h):

- objList: Danh sách các đối tượng được khai báo trong phạm vi
- owner: Đối tượng sở hữu phạm vi (program, function, procedure)
- outer: Con trỏ trả tới phạm vi bao ngoài (scope cha)

b) Symbol Table (SymTab):

- program: Đối tượng chương trình chính
- currentScope: Phạm vi hiện tại đang xử lý
- globalObjectList: Danh sách các hàm/thủ tục built-in toàn cục

3. Các loại đối tượng được quản lý

- OBJ_CONSTANT: Hằng số
- OBJ_TYPE: Kiểu dữ liệu
- OBJ_VARIABLE: Biến
- OBJ_FUNCTION: Hàm
- OBJ_PROCEDURE: Thủ tục
- OBJ_PARAMETER: Tham số
- OBJ_PROGRAM: Chương trình chính

4. Cơ chế hoạt động

a) Khai báo (Declaration):

- Khi gặp khai báo mới, kiểm tra định danh có bị trùng trong phạm vi hiện tại không (checkFreshIdent)
- Tạo đối tượng tương ứng và thêm vào danh sách của scope hiện tại

- Sử dụng hàm declareObject() để thêm đối tượng vào phạm vi

b) Tra cứu (Lookup):

- Tìm kiếm định danh từ phạm vi hiện tại lên các phạm vi ngoài
- Nếu không tìm thấy trong các phạm vi, tìm trong globalObjectList
- Sử dụng hàm lookupObject() để tìm kiếm theo cơ chế scope chain

c) Vào/Rời khỏi phạm vi:

- enterBlock(): Chuyển currentScope sang phạm vi mới khi vào block
- exitBlock(): Quay lại phạm vi ngoài khi kết thúc block

5. Ví dụ về cấu trúc scope

```

Program Main           <- Scope 1 (Program scope)
Const c = 10;          <- Khai báo trong Scope 1
Var x : Integer;       <- Khai báo trong Scope 1

Function F(...)        <- Scope 2 (Function F scope, outer = Scope 1)
  Var y : Integer;    <- Khai báo trong Scope 2
  Begin
    x := c + y;      <- x, c tra cứu từ Scope 2 lên Scope 1
  End;                 <- exitBlock() quay về Scope 1

Procedure P(...)        <- Scope 3 (Procedure P scope, outer = Scope 1)
  Const c = 20;        <- Che (shadow) hằng c của Scope 1
  Begin
    x := c;            <- c tìm thấy trong Scope 3, x tìm trong Scope 1
  End;                 <- exitBlock() quay về Scope 1

Begin
End.

```

II. Mô tả một số hàm điển hình trong Semantics.c

Hàm: Object* checkDeclaredLValueIdent(char *name)

Vị trí: semantics.c (dòng 115-140)

1. Mục đích

Kiểm tra xem một định danh có thể được sử dụng làm L-value (vẽ trái của phép gán) hay không. Đảm bảo chỉ có biến, tham số, hoặc tên hàm (để gán giá trị trả về) mới có thể được gán giá trị.

2. Các loại đối tượng hợp lệ làm L-value

- OBJ_VARIABLE: Biến thông thường

- OBJ_PARAMETER: Tham số của hàm/thủ tục
- OBJ_FUNCTION: Tên hàm (chỉ bên trong thân hàm đó để gán return value)

3. Thuật toán

Input: name - Tên định danh cần kiểm tra
 Output: Con trỏ Object nếu hợp lệ, NULL nếu có lỗi

Bước 1: Tra cứu định danh trong các phạm vi

```
obj = lookupObject(name);
```

- Sử dụng cơ chế scope chain để tìm kiếm

- Nếu obj == NULL:

 - * Báo lỗi ERR_UNDECLARED_IDENT (định danh chưa khai báo)

 - * return NULL;

Bước 2: Kiểm tra loại đối tượng

```
switch (obj->kind) {
```

- case OBJ_VARIABLE:

- case OBJ_PARAMETER:

 - > return obj (hợp lệ)

- case OBJ_FUNCTION:

 - > Kiểm tra thêm điều kiện đặc biệt (Bước 3)

- default:

 - > Báo lỗi ERR_INVALID_LVALUE

 - > return NULL

```
}
```

Bước 3: Kiểm tra đặc biệt cho OBJ_FUNCTION

- Chỉ cho phép gán cho tên hàm nếu đang trong thân hàm đó

- Kiểm tra: symtab->currentScope->owner == obj

- Nếu đúng: return obj (hợp lệ - đang gán return value)

- Nếu sai:

 - * Báo lỗi ERR_INVALID_RETURN

 - * return NULL

4. Source code thực tế

```
Object *checkDeclaredLValueIdent(char *name)
{
    Object *obj = lookupObject(name);
```

```

if (obj == NULL)
{
    error(ERR_UNDECLARED_IDENT, currentToken->lineNo, currentToken->colNo);
    return NULL;
}

switch (obj->kind)
{
case OBJ_VARIABLE:
case OBJ_PARAMETER:
    return obj;
case OBJ_FUNCTION:
    // Check if the function is the owner of the current scope
    if (symtab->currentScope->owner == obj)
        return obj;
    else
    {
        error(ERR_INVALID_RETURN, currentToken->lineNo, currentToken->colNo);
        return NULL;
    }
default:
    error(ERR_INVALID_LVALUE, currentToken->lineNo, currentToken->colNo);
    return NULL;
}
}

```

5. Ví dụ minh họa

Program Example;

Var x : Integer; <- Scope 1 (Program)

Function Sum(a: Integer) : Integer; <- Scope 2 (Function Sum)

 Var temp : Integer; <- Khai báo trong Scope 2

 Begin

 temp := a + 10; <- temp, a hợp lệ (OBJ_VARIABLE, OBJ_PARAMETER)

 Sum := temp; <- Sum hợp lệ (owner của Scope 2)

 End;

Begin

 x := Sum(5); <- x hợp lệ (OBJ_VARIABLE trong Scope 1)

 Sum := 100; <- LỖI! Sum không phải owner của Scope 1

 -> ERR_INVALID_RETURN

End.

Giai thích từng trường hợp:

a) temp := a + 10;

- Tra cứu temp: Tìm thấy trong Scope 2, kind = OBJ_VARIABLE
- Kết quả: PASS (biến hợp lệ làm L-value)

b) Sum := temp;

- Tra cứu Sum: Tìm thấy, kind = OBJ_FUNCTION
- Kiểm tra: symtab->currentScope->owner == Sum? YES (đang trong thân Sum)
- Kết quả: PASS (gán giá trị trả về cho hàm)

c) x := Sum(5);

- Tra cứu x: Tìm thấy trong Scope 1, kind = OBJ_VARIABLE
- Kết quả: PASS (biến hợp lệ)

d) Sum := 100;

- Tra cứu Sum: Tìm thấy, kind = OBJ_FUNCTION
- Kiểm tra: symtab->currentScope->owner == Sum? NO (đang ở Program scope)
- Kết quả: ERR_INVALID_RETURN (không thể gán cho hàm từ bên ngoài)

III. Kết quả thực hiện với EXAMPLE

example 4:

```
PROGRAM EXAMPLE4; (* Example 4 *)
  CONST MAX = 10;
  TYPE T = Int;
  Var A : Arr[10,Int];
  Var N : Int;
  Var CH : Char;
  Procedure INPUT;
    Var I ; Int;
    Var TMP : Int;
  BEGIN
    N := READI;
    FOR I := 1 TO N DO
      A(I,.) := READI;
  END;
  Procedure OUTPUT;
    Var I : Int;
  BEGIN
    FOR I := 1 TO N DO
      BEGIN
        CALL WRITEI(A(I,.));
      END;
  END;
  Function SUM : Int;
    Var I : Int;
    Var S : Int;
  BEGIN
    S := 0;
    FOR I := 1 TO N DO
      S := S + A(I,);
    RETURN S;
  END;
END;
```

IV. Kết quả thực hiện với các ERROR ERR_UNDECLARED_IDENT

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists several KPL files under 'tests/error_cases'. The main editor area contains the following KPL code:

```
1 PROGRAM ERR1;
2 CONST MAX = 10;
3 VAR N : INTEGER;
4 BEGIN
5   N := Y;
6 END.
```

The terminal below the editor shows two error messages:

```
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./tests/error_cases/err1.kpl
5-9:ERR_UNDECLARED_IDENT
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %
```

Code:

```
N := Y;
```

Lỗi:

Identifier 'Y' chưa được khai báo trong bất kỳ scope nào.

Nguyên nhân:

- 'Y' được sử dụng trong expression nhưng không có khai báo
- Compiler không tìm thấy 'Y' trong scope hiện tại và các outer scopes

Hàm xử lý:

1. compileFactor() -> gấp TK_IDENTIFIER 'Y'
2. Gọi checkDeclaredIdent("Y") (semantics.c:38)
3. checkDeclaredIdent() gọi lookupObject("Y") -> return NULL
4. Phát hiện obj == NULL -> gọi error(ERR_UNDECLARED_IDENTIFIER, ...)

Flow: compileFactor() -> checkDeclaredIdent() -> lookupObject() -> error()

ERR_UNDECLARED_CONSTANT

```

Scope_Management_incompleted
example4.kpl tests err2.kpl
tests > error_cases > err2.kpl
1 PROGRAM ERR2;
2 CONST
3 MAX = 10;
4 MIN = NOCONST;
5 BEGIN
6 END.
7

```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE DEVDB

- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err1.kpl
 5-9:ERR_UNDECLARED_IDENTIFIER
- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err2.kpl
 4-9:ERR_UNDECLARED_CONSTANT
- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

Code:

```

CONST
MAX = 10;
MIN = NOCONST;

```

Lỗi:

Identifier 'NOCONST' chưa được khai báo làm constant.

Nguyên nhân:

- Trong khai báo constant, giá trị phải là số, ký tự, hoặc constant khác
- 'NOCONST' không tồn tại trong symbol table

Hàm xử lý:

1. compileBlock() -> gấp CONST declaration
2. compileConstant() -> compileUnsignedConstant() -> gấp TK_IDENT
3. Gọi checkDeclaredConstant("NOCONST") (semantics.c:46)
4. checkDeclaredConstant() gọi lookupObject("NOCONST") -> return NULL
5. Phát hiện obj == NULL -> gọi error(ERR_UNDECLARED_CONSTANT, ...)

Flow: compileConstant() -> checkDeclaredConstant() -> lookupObject() -> error()

ERR_UNDECLARED_TYPE

The screenshot shows a code editor interface with the following details:

- Explorer View:** Shows a file tree with several KPL files (e.g., err1.kpl, err2.kpl, err3.kpl, err4.kpl, etc.) and C/C++ header files (charcode.c, charcode.h, debug.c, debug.h, error.c, error.h, error.o, kpl, main.c, parser.c).
- Editor View:** The current file is `err3.kpl`. The code contains the following KPL code:


```

1 PROGRAM ERR3;
2 CONST MAX = 10;
3 TYPE T = INTEGER;
4 VAR A : T2;
5 BEGIN
6 END.
    
```
- Terminal View:** Shows the output of running the script:


```

duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./tests/error_cases/err3.kpl
4-9:ERR_UNDECLARED_TYPE
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %
    
```

Code:

```
VAR A : T2;
```

Lỗi:

Type 'T2' chưa được khai báo.

Nguyên nhân:

- Trong khai báo biến, type 'T2' không tồn tại
- Chỉ có type 'T' = INTEGER được khai báo

Hàm xử lý:

1. `compileVarDecl()` -> `compileType()` -> gấp TK_IDENT 'T2'
2. Gọi `checkDeclaredType("T2")` (semantics.c:62)
3. `checkDeclaredType()` gọi `lookupObject("T2")` -> return NULL
4. Phát hiện obj == NULL -> gọi `error(ERR_UNDECLARED_TYPE, ...)`

Flow: `compileType()` -> `checkDeclaredType()` -> `lookupObject()` -> `error()`

ERR_UNDECLARED_VARIABLE

```

Scope_Management_incompleted
err4.kpl

tests > error_cases > err4.kpl
1 PROGRAM ERR4;
2 CONST MAX = 10;
3 VAR I : INTEGER;
4 BEGIN
5   FOR Z := 1 TO MAX DO
6     I := I + 1;
7 END.
8

```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE DEVDB

- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err3.kpl
 4-9:ERR_UNDECLARED_TYPE
 5-8:ERR_UNDECLARED_VARIABLE
 8:duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

Code:

FOR Z := 1 TO MAX DO

Lỗi:

Variable 'Z' chưa được khai báo.

Nguyên nhân:

- FOR loop yêu cầu control variable phải được khai báo trước
- 'Z' không có trong khai báo VAR

Hàm xử lý:

1. compileForSt() -> gấp TK_IDENT 'Z'
2. Gọi checkDeclaredVariable("Z") (semantics.c:77)
3. checkDeclaredVariable() gọi lookupObject("Z") -> return NULL
4. Phát hiện obj == NULL -> gọi error(ERR_UNDECLARED_VARIABLE, ...)

Flow: compileForSt() -> checkDeclaredVariable() -> lookupObject() -> error()

ERR_INVALID_VARIABLE

```

Scope_Management_incompleted
err5.kpl
tests > error_cases > err5.kpl
1 PROGRAM ERR5;
2 CONST MAX = 10;
3 VAR I : INTEGER;
4 BEGIN
5   FOR MAX := 1 TO 10 DO
6     I := I + 1;
7 END.
8

```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE DEVDB

- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err5.kpl
 5-8:ERR_INVALID_VARIABLE
 duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

Code:

```
FOR MAX := 1 TO 10 DO
```

Lỗi:

'MAX' là constant, không phải variable.

Nguyên nhân:

- FOR loop yêu cầu control variable là biến có thể gán
- 'MAX' được khai báo là CONST, không thể thay đổi giá trị

Hàm xử lý:

1. compileForSt() -> gấp TK_IDENT 'MAX'
2. Gọi checkDeclaredVariable("MAX") (semantics.c:77)
3. checkDeclaredVariable() gọi lookupObject("MAX") -> tìm thấy obj
4. Kiểm tra obj->kind != OBJ_VARIABLE -> gọi error(ERR_INVALID_VARIABLE, ...)

Flow: compileForSt() -> checkDeclaredVariable() -> error()

ERR_INVALID_RETURN

```

PROGRAM ERRE;
FUNCTION F : INTEGER;
VAR N : INTEGER;
BEGIN
N := 10;
END;
BEGIN
F := 20;
END.

```

Code:

```

FUNCTION F : INTEGER;
BEGIN
N := 10;
END;
BEGIN
F := 20;    <- Lỗi ở đây
END.

```

Lỗi:

Gán giá trị cho function 'F' từ bên ngoài thân hàm.

Nguyên nhân:

- Chỉ được phép gán cho tên hàm bên trong thân hàm đó (để set return value)
- Ở đây đang gán 'F := 20' từ program scope, không phải từ function F scope

Hàm xử lý:

1. compileLValue() -> gấp TK_IDENT 'F'
2. Gọi checkDeclaredLValueIdent("F") (semantics.c:115)
3. checkDeclaredLValueIdent() gọi lookupObject("F") -> tìm thấy obj (OBJ_FUNCTION)
4. Kiểm tra: obj->kind == OBJ_FUNCTION
5. Kiểm tra thêm: symtab->currentScope->owner == obj? -> FALSE
(currentScope->owner là PROGRAM, không phải F)

6. Gọi error(ERR_INVALID_RETURN, ...)

Flow: compileLValue() -> checkDeclaredLValueIdent() -> error()

ERR_UNDECLARED_PROCEDURE

The screenshot shows a code editor interface with the following details:

- Explorer View:** Shows a project structure with files like example4.kpl, err7.kpl, and various error cases (err1.kpl to err13.kpl).
- Editor View:** Displays the content of err7.kpl:

```
1 PROGRAM ER6;
2 CONST MAX = 10;
3 VAR N : INTEGER;
4 BEGIN
5   CALL P(N);
6 END.
```
- Terminal View:** Shows the output of running the program:

```
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err7.kpl
5-9:ERR_UNDECLARED_PROCEDURE
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %
```
- Bottom Status Bar:** Includes information like Ln 1, Col 1, Spaces: 3, UTF-8, LF, Plain Text, Go Live, Colorize: 0 variables, Colorize, and Prettier.

Code:

```
CALL P(N);
```

Lỗi:

Procedure 'P' chưa được khai báo.

Nguyên nhân:

- Lệnh CALL yêu cầu procedure phải được khai báo trước
- 'P' không tồn tại trong symbol table

Hàm xử lý:

1. compileCallSt() -> gấp KW_CALL và TK_IDENT 'P'
2. Gọi checkDeclaredProcedure("P") (semantics.c:99)
3. checkDeclaredProcedure() gọi lookupObject("P") -> return NULL
4. Phát hiện obj == NULL -> gọi error(ERR_UNDECLARED_PROCEDURE, ...)

Flow: compileCallSt() -> checkDeclaredProcedure() -> lookupObject() -> error()

ERR_DUPLICATE_IDENT

```

PROGRAM ERR7;
CONST MAX = 10;
PROCEDURE P(N:INTEGER; N:CHAR);
BEGIN
END;
BEGIN
END.

```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE DEVDB

- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err7.kpl
 - 5-9:ERR_UNDECLARED_PROCEDURE
 - 3-24:ERR_DUPLICATE_IDENT
- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

Code:

```
PROCEDURE P(N:INTEGER; N:CHAR);
```

Lỗi:

Parameter 'N' được khai báo 2 lần.

Nguyên nhân:

- Trong cùng một scope (parameter list của procedure P)
- Identifier 'N' xuất hiện 2 lần với type khác nhau

Hàm xử lý:

1. compileParam() -> gấp TK_IDENT 'N' lần đầu
2. Gọi checkFreshIdent("N") (semantics.c:33) -> PASS (chưa có)
3. Khai báo parameter N:INTEGER thành công
4. Gấp TK_IDENT 'N' lần 2
5. Gọi checkFreshIdent("N") -> tìm thấy 'N' trong scope hiện tại
6. Gọi error(ERR_DUPLICATE_IDENT, ...)

Flow: compileParam() -> checkFreshIdent() -> findObject() -> error()

ERR_INVALID_LVALUE

The screenshot shows a code editor interface with the following details:

- Explorer:** Shows a project structure with files like `example4.kpl`, `err9.kpl`, and various `errX.kpl` files under `tests/error_cases`.
- Editor:** The current file is `err9.kpl` which contains the following KPL code:

```

1 PROGRAM ERRB;
2 CONST MAX = 10;
3 VAR N : INTEGER;
4 BEGIN
5   MAX := 20;
6 END.

```
- Terminal:** Shows command-line output from running the file:

```

duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err9.kpl
5-9:ERR_UNDECLARED_PROCEDURE
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err8.kpl
3-24:ERR_IDENTIFIER_EXPECTED
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err9.kpl
5-4:ERR_INVALID_LVALUE
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

```
- Bottom Status Bar:** Shows the terminal prompt `duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %`.

Code:

```
MAX := 20;
```

Lỗi:

'MAX' là constant, không thể làm L-value (vẽ trái phép gán).

Nguyên nhân:

- Assignment yêu cầu L-value phải là: variable, parameter, hoặc function name
- 'MAX' là OBJ_CONSTANT, không nằm trong các loại hợp lệ

Hàm xử lý:

1. `compileLValue() ->` gặp TK_IDENT 'MAX'
2. Gọi `checkDeclaredLValueIdent("MAX")` (semantics.c:115)
3. `checkDeclaredLValueIdent()` gọi `lookupObject("MAX") ->` tìm thấy obj
4. Kiểm tra `switch(obj->kind):`
 - case OBJ_VARIABLE: không match
 - case OBJ_PARAMETER: không match
 - case OBJ_FUNCTION: không match
 - default: match
5. Gọi `error(ERR_INVALID_LVALUE, ...)

Flow: `compileLValue() -> checkDeclaredLValueIdent() -> error()

ERR_INVALID_CONSTANT

```

Scope_Management_incompleted
err10.kpl
tests > error_cases > err10.kpl
1 PROGRAM ERR9;
2 VAR N : INTEGER;
3 PROCEDURE P;
4 CONST MAX = N;
5 BEGIN
6 END;
7 BEGIN
8 END.
9

```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE DEVDB

duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err10.kpl
4-13:ERR_INVALID_CONSTANT
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

Code:

```
VAR N : MAX;
```

Lỗi:

'MAX' là constant (số), không phải type.

Nguyên nhân:

- Trong khai báo biến, sau dấu ':' phải là type
- 'MAX' được tìm thấy nhưng là OBJ_CONSTANT, không phải OBJ_TYPE

Hàm xử lý:

1. compileVarDecl() -> compileType() -> gặp TK_IDENT 'MAX'
2. Gọi checkDeclaredType("MAX") (semantics.c:62)
3. checkDeclaredType() gọi lookupObject("MAX") -> tìm thấy obj
4. Kiểm tra obj->kind != OBJ_TYPE -> gọi error(ERR_INVALID_TYPE, ...)

Chú ý: Error code là ERR_INVALID_TYPE, nhưng message in ra là "A type expected" (đây là syntax error code được dùng cho semantic check)

Flow: compileType() -> checkDeclaredType() -> error()

ERR_INVALID_TYPE

The screenshot shows a code editor interface with a dark theme. In the top right, there are icons for BLACKBOX, PURECODE AI, and other tools. The left sidebar is titled 'EXPLORER' and shows a tree view of files. The main editor area has tabs for 'example4.kpl' and 'err11.kpl'. The code in 'err11.kpl' is as follows:

```

1 PROGRAM ERR10;
2 CONST MAX = 10;
3 VAR N : MAX;
4 BEGIN
5 END.
6

```

The terminal tab at the bottom shows the following output:

```

duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err10.kpl
4-13:ERR_INVALID_CONSTANT
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err11.kpl
3-9:ERR_INVALID_TYPE
duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

```

Code:

$N := T + 5;$

Lỗi:

'T' là type, không thể dùng trong expression.

Nguyên nhân:

- Expression yêu cầu các operand là value (variable, constant, function call)
- 'T' là OBJ_TYPE, không có giá trị để tính toán

Hàm xử lý:

1. compileFactor() -> gặt TK_IDENT 'T'
2. Gọi checkDeclaredIdent("T") (semantics.c:38) -> tìm thấy obj
3. Kiểm tra switch(obj->kind):
 - case OBJ_CONSTANT: không match
 - case OBJ_VARIABLE: không match
 - case OBJ_PARAMETER: không match
 - case OBJ_FUNCTION: không match
 - default: match (T là OBJ_TYPE)
4. Gọi error(ERR_INVALID_FACTOR, ...)

Chú ý: Message là "Invalid factor" - đây là syntax error được dùng cho semantic

Flow: compileFactor() -> checkDeclaredIdent() -> error()

ERR_INVALID_FACTOR

```
PROGRAM ERR11;
CONST MAX = 10;
TYPE T = INTEGER;
VAR N : INTEGER;
BEGIN
    N := T + 5;
END.
```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE DEVDB

duongcongthuyet@Dungs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err12.kpl
6-9:ERR_INVALID_FACTOR
duongcongthuyet@Dungs-MacBook-Pro Scope_Management_incompleted %

Code:

```
CALL SUM;
```

Lỗi:

'SUM' là function, không phải procedure.

Nguyên nhân:

- CALL statement chỉ dùng cho procedure
- 'SUM' được khai báo là FUNCTION, không phải PROCEDURE

Hàm xử lý:

1. compileCallSt() -> gặp KW_CALL và TK_IDENT 'SUM'
2. Gọi checkDeclaredProcedure("SUM") (semantics.c:99)
3. checkDeclaredProcedure() gọi lookupObject("SUM") -> tìm thấy obj
4. Kiểm tra obj->kind != OBJ_PROCEDURE -> gọi error(ERR_INVALID_PROCEDURE, ...)

Chú ý: Error code là ERR_INVALID_PROCEDURE, message "A procedure identifier expected"

Flow: compileCallSt() -> checkDeclaredProcedure() -> error()

ERR_INVALID_PROCEDURE

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists several KPL files under 'OPEN EDITORS' and 'SCOPE MANAGEMENT_INCOMPLETED'. The 'tests' folder contains sub-folders like 'error_cases' and files such as 'err1.kpl' through 'err13.kpl'. The main editor area displays the content of 'err13.kpl':

```
1 PROGRAM ERR12;
2 CONST MAX = 10;
3 FUNCTION SUM : INTEGER;
4 BEGIN
5   SUM := MAX;
6 END;
7 BEGIN
8   CALL SUM;
9 END.
10
```

Below the code editor is the 'PROBLEMS' tab of the 'Terminal' panel, showing three errors:

- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err12.kpl
- 6-9:ERR_INVALID_FACTOR
- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted % ./kpl ./tests/error_cases/err13.kpl
- 8-9:ERR_INVALID_PROCEDURE
- duongcongthuyet@Duongs-MacBook-Pro Scope_Management_incompleted %

At the bottom of the interface, there are various status indicators and toolbars.

Code:

```
CALL SUM;
```

Lỗi:

Identifier không phải procedure.

Hàm xử lý:

Giống err12 - checkDeclaredProcedure() phát hiện obj->kind != OBJ PROCEDURE