

Báo cáo LAB06
IT3323 Mã lớp 161269
Bài 6 : Type checking part II

Họ và tên: Dương Công Thuyết

MSSV: 20225932

- I. Mô tả các hàm `compileArgument` và `compileArguments` trong `parser.c` và ứng dụng của hàm đó trong kiểm tra tương ứng kiểu giữa khai báo và sử dụng hàm, thủ tục

HÀM `compileArgument()`

1. Chức năng

Hàm `compileArgument()` có nhiệm vụ phân tích cú pháp và kiểm tra kiểu của một đối số khi gọi hàm/thủ tục, đảm bảo tính tương ứng về kiểu và loại (giá trị/ tham chiếu) với tham số đã khai báo.

2. Cơ chế hoạt động

Hàm xử lý hai trường hợp chính:

Trường hợp 1: Tham số giá trị (`PARAM_VALUE`)

```
if (param->paramAttrs->kind == PARAM_VALUE) {  
    argType = compileExpression();  
    checkTypeEquality(argType, param->paramAttrs->type);  
}
```

- Cho phép truyền biểu thức hoặc giá trị
- Gọi `compileExpression()` để lấy kiểu của đối số
- Gọi `checkTypeEquality(argType, param->paramAttrs->type)` để so sánh kiểu
- Nếu kiểu không khớp → báo lỗi `ERR_TYPE_INCONSISTENCY`

Trường hợp 2: Tham số tham chiếu (`PARAM_REFERENCE`)

```
else {  
    argType = compileLValue();  
    checkTypeEquality(argType, param->paramAttrs->type);  
}
```

- Chỉ cho phép truyền lvalue (biến, phần tử mảng, tham số)

- Gọi `compileLValue()` để lấy kiểu của `lvalue`
- Gọi `checkTypeEquality(argType, param->paramAttrs->type)` để so sánh kiểu
- Nếu không phải `lvalue` hoặc kiểu không khớp → báo lỗi

3. Ứng dụng trong kiểm tra kiểu

Quy tắc kiểm tra:

Hàm áp dụng hai quy tắc chính:

1. Tham số giá trị: Cho phép bất kỳ biểu thức nào có kiểu khớp
2. Tham số tham chiếu (VAR): Chỉ cho phép `lvalue` có kiểu khớp

Vị trí sử dụng:

Được gọi trong hàm `compileArguments()` khi xử lý từng đối số:

```
compileArgument(node->object); // node->object là tham số tương ứng
```

Ý nghĩa:

Hàm đảm bảo tính an toàn kiểu khi truyền đối số cho hàm/thủ tục. Việc phân biệt giữa tham số giá trị và tham số tham chiếu giúp ngăn chặn các lỗi như truyền hằng số hoặc biểu thức cho tham số VAR - một thao tác không hợp lệ vì tham số VAR cần địa chỉ biến để có thể thay đổi giá trị. Kiểm tra được thực hiện thông qua `checkTypeEquality()` và báo lỗi `ERR_TYPE_INCONSISTENCY` ngay lập tức khi phát hiện vi phạm.

HÀM `compileArguments()`

1. Chức năng

Hàm `compileArguments()` có nhiệm vụ phân tích cú pháp và kiểm tra toàn bộ danh sách đối số khi gọi hàm/thủ tục, đảm bảo tính tương ứng về số lượng, thứ tự và kiểu với danh sách tham số đã khai báo.

2. Cơ chế hoạt động

Hàm xử lý bốn trường hợp chính:

Trường hợp 1: Bắt đầu danh sách đối số

case `SB_LPAR`:

```

eat(SB_LPAR);
if (node == NULL)
    error(ERR_PARAMETERS_ARGUMENTS_INCONSISTENCY, currentToken->lineNo,
currentToken->colNo);
compileArgument(node->object);
node = node->next;

```

- Gặp dấu mở ngoặc (
- Kiểm tra node == NULL: Nếu đúng → thừa đối số (không còn tham số để khớp)
- Gọi compileArgument() để xử lý đối số đầu tiên
- Di chuyển con trỏ node đến tham số tiếp theo

Trường hợp 2: Xử lý các đối số tiếp theo

```

while (lookAhead->tokenType == SB_COMMA) {
    eat(SB_COMMA);
    if (node == NULL)
        error(ERR_PARAMETERS_ARGUMENTS_INCONSISTENCY, currentToken->lineNo,
currentToken->colNo);
    compileArgument(node->object);
    node = node->next;
}

```

- Gặp dấu phẩy , → có đối số tiếp theo
- Kiểm tra node == NULL: Nếu đúng → thừa đối số
- Gọi compileArgument() cho từng đối số
- Di chuyển con trỏ node sau mỗi đối số

Trường hợp 3: Kết thúc danh sách đối số

```

if (node != NULL)
    error(ERR_PARAMETERS_ARGUMENTS_INCONSISTENCY, currentToken->lineNo,
currentToken->colNo);

```

```

eat(SB_RPAR);

```

- Kiểm tra node != NULL: Nếu đúng → thiếu đối số (còn tham số chưa được truyền)
- Gặp dấu đóng ngoặc) → kết thúc

Trường hợp 4: Không có đối số

default:

`break; // Hàm không có tham số`

- Hàm/thủ tục được gọi không có dấu ngoặc
- Kiểm tra `paramList == NULL` ngầm định (không có tham số khai báo)

3. Ứng dụng trong kiểm tra kiểu

Quy tắc kiểm tra:

Hàm áp dụng ba quy tắc chính:

1. Số lượng: Số đối số phải bằng số tham số
2. Thứ tự: Đối số thứ *i* phải khớp với tham số thứ *i*
3. Kiểu: Mỗi đối số phải có kiểu tương ứng với tham số (thông qua `compileArgument()`)

Vị trí sử dụng:

Được gọi trong hai hàm:

1. `compileCallSt()` - Khi gọi thủ tục:

```
proc = checkDeclaredProcedure(currentToken->string);
compileArguments(proc->procAttrs->paramList);
```

2. `compileFactor()` - Khi gọi hàm trong biểu thức:

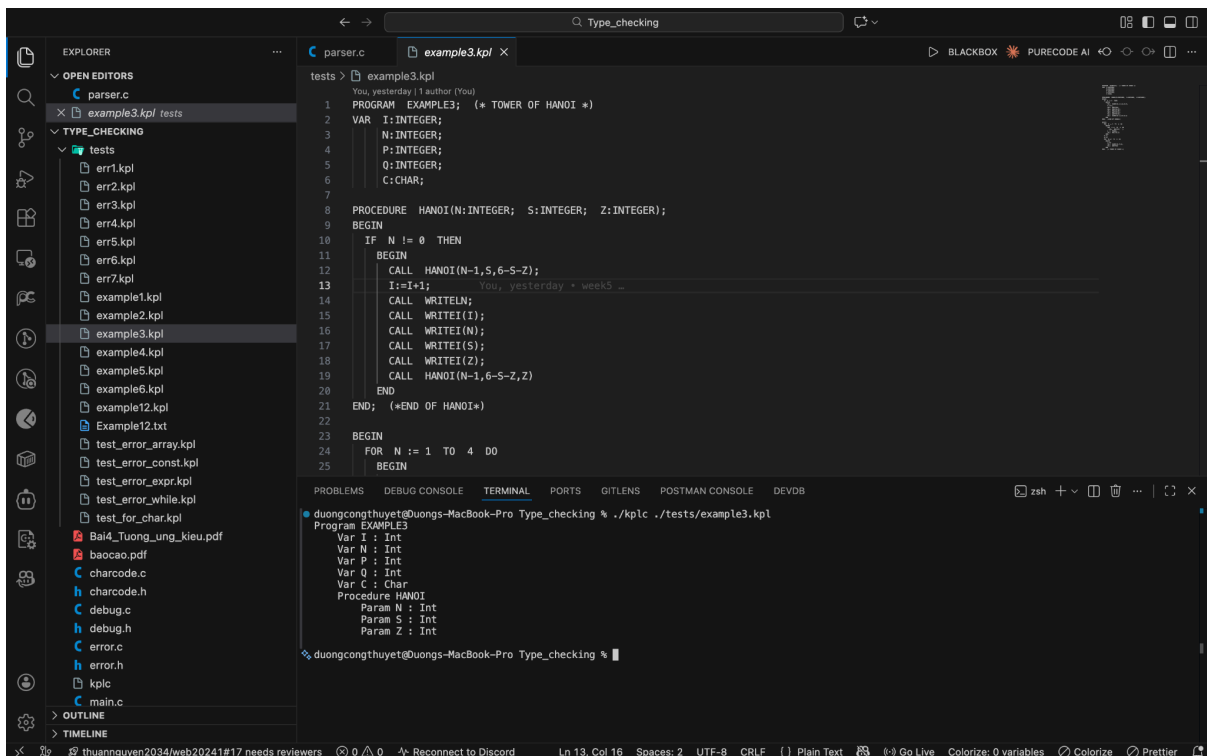
```
compileArguments(obj->funcAttrs->paramList);
```

Ý nghĩa:

Hàm đảm bảo tính nhất quán giữa khai báo và sử dụng hàm/thủ tục. Việc kiểm tra số lượng đối số (qua điều kiện `node == NULL` và `node != NULL`) và kiểm tra từng đối số (qua `compileArgument()`) giúp phát hiện sớm các lỗi truyền tham số sai - một nguồn lỗi phổ biến trong lập trình. Hàm báo lỗi

`ERR_PARAMETERS_ARGUMENTS_INCONSISTENCY` khi số lượng không khớp và `ERR_TYPE_INCONSISTENCY` (qua `compileArgument()`) khi kiểu không khớp, đảm bảo tính an toàn kiểu trong toàn bộ chương trình.

II. Kết quả thực hiện với EXAMPLE3.KPL không có lỗi



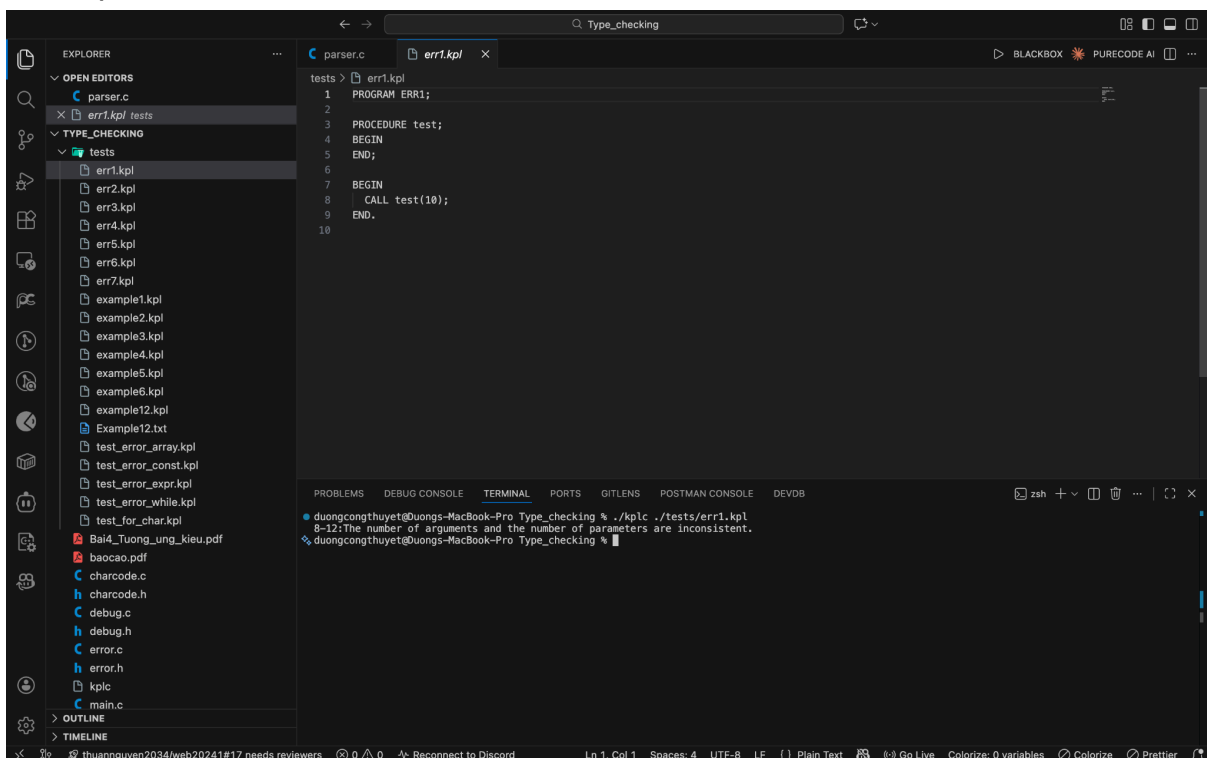
```
1 PROGRAM EXAMPLE3; (* TOWER OF HANOI *)
2 VAR I: INTEGER;
3     N: INTEGER;
4     P: INTEGER;
5     Q: INTEGER;
6     C: CHAR;
7
8 PROCEDURE HANOI(N: INTEGER; S: INTEGER; Z: INTEGER);
9 BEGIN
10 IF N != 0 THEN
11 BEGIN
12 CALL HANOI(N-1, S, 6-S-Z);
13 I:=I+1; You, yesterday + week5 ...
14 CALL WRITELN;
15 CALL WRITEI(I);
16 CALL WRITEI(N);
17 CALL WRITEI(S);
18 CALL WRITEI(Z);
19 CALL HANOI(N-1, 6-S-Z, Z);
20 END;
21 END; (*END OF HANOI*)
22
23 BEGIN
24 FOR N := 1 TO 4 DO
25 BEGIN
```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLINS POSTMAN CONSOLE DEVDB

```
duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./kplc ./tests/example3.kpl
Program EXAMPLE3
Var I : Int
Var N : Int
Var P : Int
Var Q : Int
Var C : Char
Procedure HANOI
Param N : Int
Param S : Int
Param Z : Int
```

III. Thay đổi trong các ví dụ đã cho để gây các lỗi:

1. Danh sách tham số hình thức rỗng, danh sách tham số thực sự có ít nhất 1 phần tử



```
1 PROGRAM ERR1;
2
3 PROCEDURE test;
4 BEGIN
5 END;
6
7 BEGIN
8 CALL test(10);
9 END.
```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLINS POSTMAN CONSOLE DEVDB

```
duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./kplc ./tests/err1.kpl
8-12: The number of arguments and the number of parameters are inconsistent.
```

Lỗi: 8-12: The number of arguments and the number of parameters are inconsistent.

Đoạn code gây lỗi:

```
PROCEDURE test;  
BEGIN  
END;
```

```
BEGIN
```

```
CALL test(10); // Lỗi: test không có tham số nhưng truyền đối số 10  
END.
```

Nguyên nhân:

- Procedure test không có tham số khai báo (paramList = NULL)
- Gọi CALL test(10) với 1 đối số

Hàm phát hiện lỗi: compileArguments()

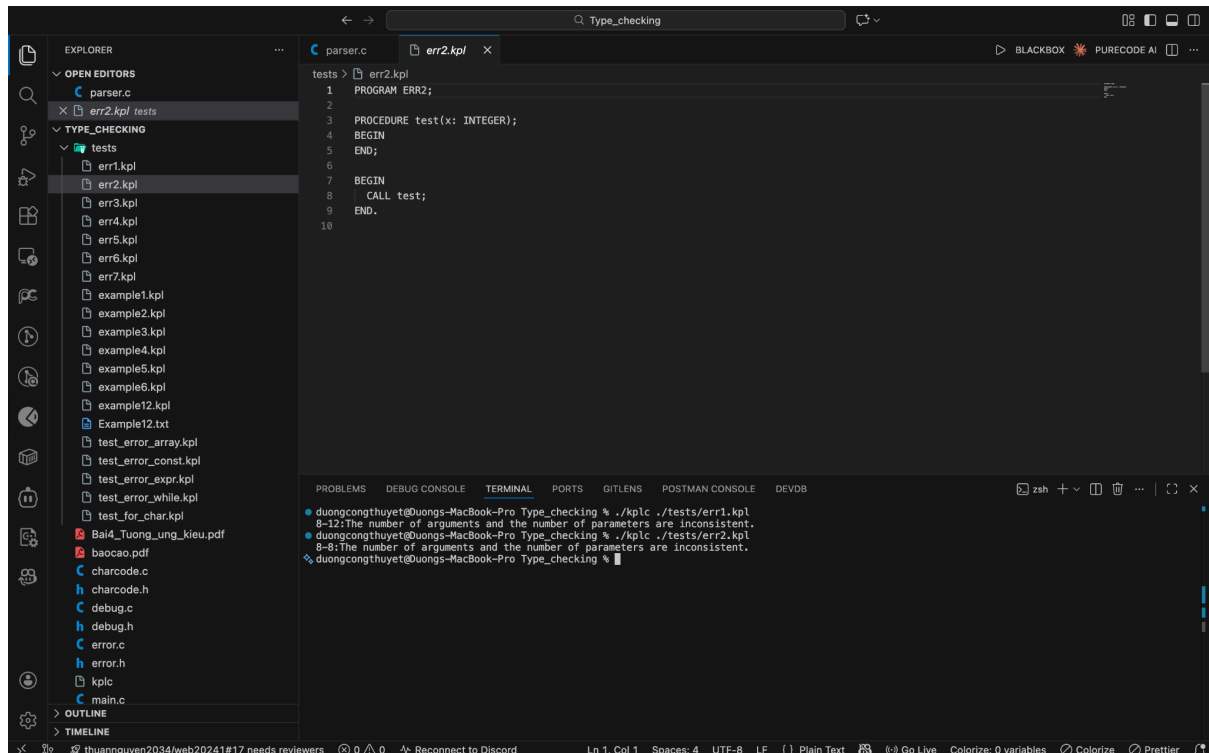
- Khi gặp SB_LPAR và node == NULL

- Điều kiện: if (node == NULL)

error(ERR_PARAMETERS_ARGUMENTS_INCONSISTENCY)

- Vị trí: parser.c, dòng ~593

2. Danh sách tham số thực sự rỗng, danh sách tham số hình thức có ít nhất 1 phần tử



Lỗi: 8-8: The number of arguments and the number of parameters are inconsistent.

Đoạn code gây lỗi:

```
PROCEDURE test(x: INTEGER);  
BEGIN  
END;
```

```
BEGIN  
    CALL test;      // Lỗi: test có tham số x nhưng không truyền đối số  
END.
```

Nguyên nhân:

- Procedure test có 1 tham số: x: INTEGER
- Gọi CALL test; không truyền đối số (không có dấu ngoặc)

Hàm phát hiện lỗi: compileArguments()

- Khi không gặp SB_LPAR (FOLLOW set) nhưng node != NULL

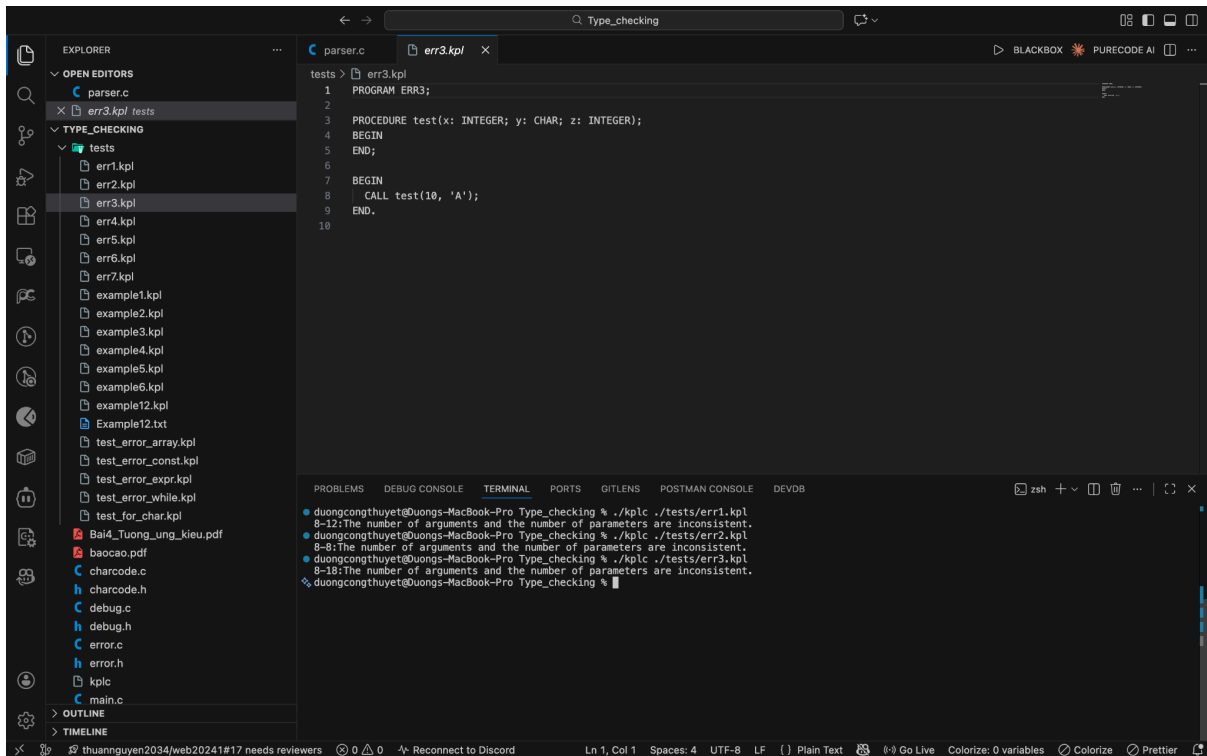
Đoạn code gây lỗi:

```
PROCEDURE test(x: INTEGER; y: CHAR; z: INTEGER);  
BEGIN  
END;
```

```
BEGIN  
    CALL test(10, 'A'); // Lỗi: test có 3 tham số nhưng chỉ truyền 2 đối số  
END.
```

- Điều kiện: if (node != NULL)
error(ERR_PARAMETERS_ARGUMENTS_INCONSISTENCY)
- Vị trí: parser.c, dòng ~629 (trong case SB_SEMICOLON và FOLLOW set khác)

3. Danh sách tham số hình thức và tham số thực sự thực sự chứa tối thiểu 1 phần tử nhưng danh sách tham số hình thức chứa nhiều phần tử hơn danh sách tham số thực sự



Lỗi: 8-18:The number of arguments and the number of parameters are inconsistent.

Nguyên nhân:

- Procedure test có 3 tham số: x, y, z
- Gọi CALL test(10, 'A') chỉ truyền 2 đối số

Hàm phát hiện lỗi: compileArguments()

- Sau khi xử lý hết đối số, gập SB_RPAR nhưng node != NULL

Đoạn code gây lỗi:

```
PROCEDURE test(x: INTEGER; y: CHAR);  
BEGIN  
END;
```

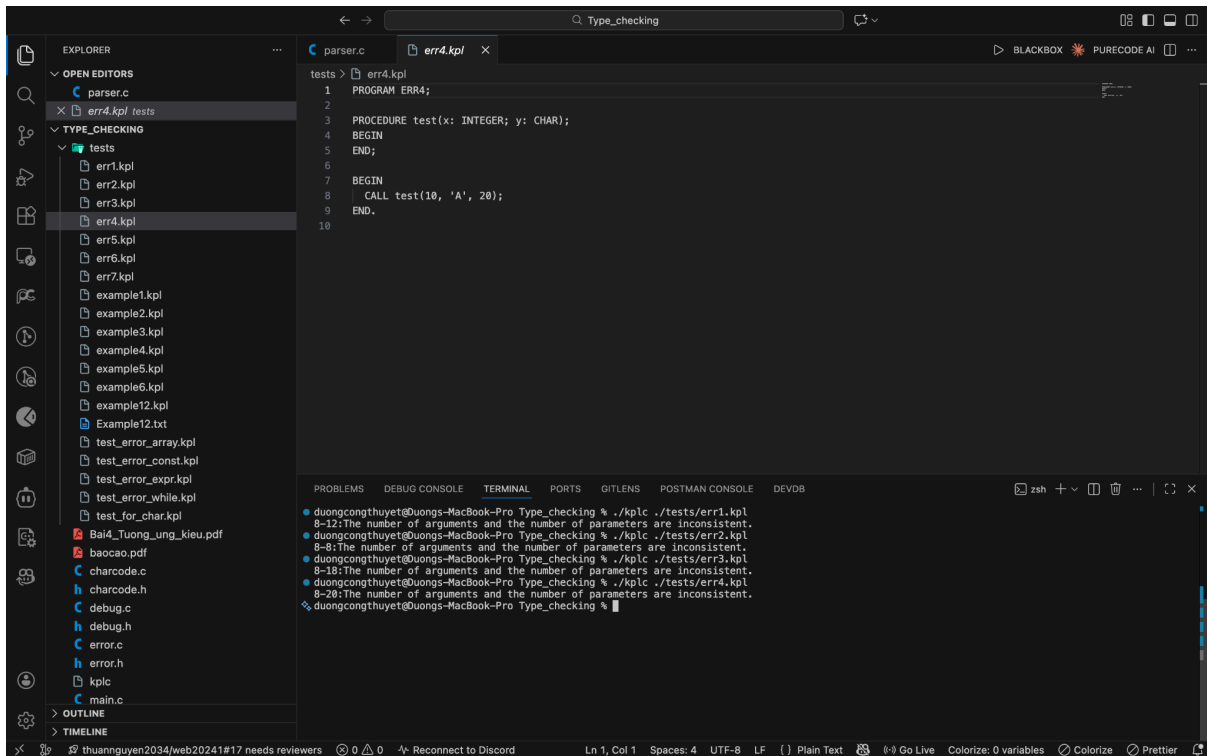
```
BEGIN
```

```
    CALL test(10, 'A', 20);    // Lỗi: test có 2 tham số nhưng truyền 3 đối số  
END.
```

- Điều kiện: if (node != NULL)
- ```
error(ERR_PARAMETERS_ARGUMENTS_INCONSISTENCY)
```
- Vị trí: parser.c, dòng ~609 (trước eat(SB\_RPAR))

4. Danh sách tham số hình thức và tham số thực sự thực sự chứa tối thiểu 1 phần tử nhưng danh sách tham số thực sự chứa nhiều phần tử hơn danh sách tham số hình thức





Lỗi: 8-20:The number of arguments and the number of parameters are inconsistent.

Nguyên nhân:

- Procedure test có 2 tham số: x, y
- Gọi CALL test(10, 'A', 20) truyền 3 đối số

Hàm phát hiện lỗi: compileArguments()

- Khi gặp SB\_COMMA nhưng node == NULL (hết tham số để khớp)

Đoạn code gây lỗi:

```

VAR x: INTEGER;
 c: CHAR;

```

```

PROCEDURE test(a: INTEGER; b: CHAR);
BEGIN
END;

```

```

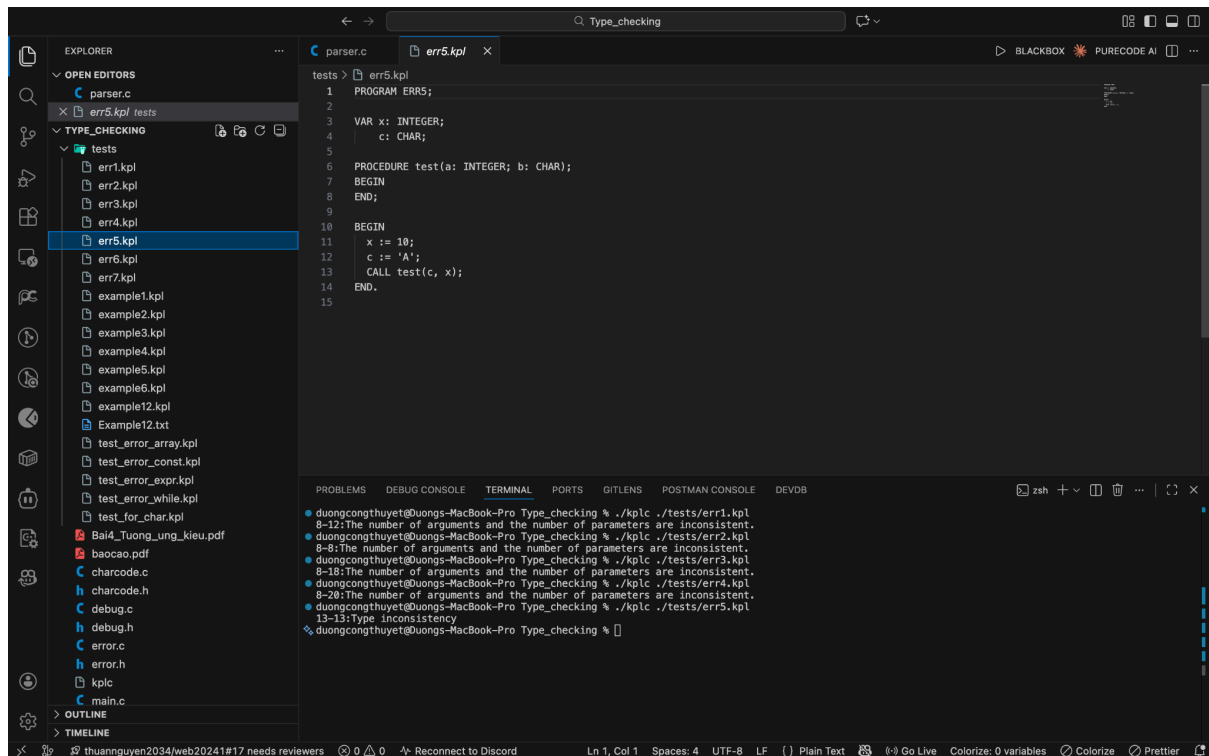
BEGIN
 x := 10;
 c := 'A';
 CALL test(c, x); // Lỗi: truyền CHAR cho INTEGER, INTEGER cho CHAR
END.

```

- Điều kiện: if (node == NULL)
- error(ERR\_PARAMETERS\_ARGUMENTS\_INCONSISTENCY)

- Vị trí: parser.c, dòng ~600 (trong vòng lặp while SB\_COMMA)

## 5. Không tương ứng kiểu giữa tham số hình thức và tham số thực sự



Lỗi: 13-13:Type inconsistency

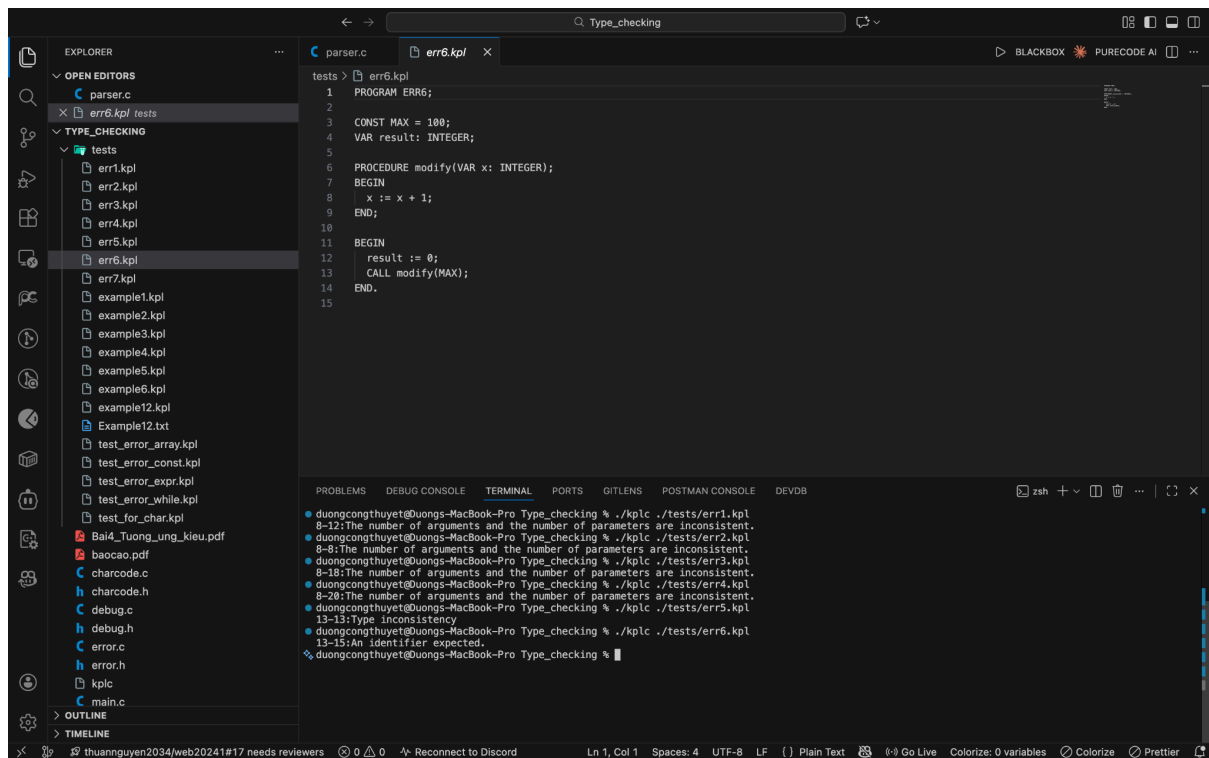
Nguyên nhân:

- Procedure test có tham số: a: INTEGER, b: CHAR
- Gọi CALL test(c, x) với c là CHAR, x là INTEGER
- Tham số đầu tiên sai kiểu: CHAR != INTEGER

Hàm phát hiện lỗi: `compileArgument()` -> `checkTypeEquality()`

- `compileArgument()` gọi `compileExpression()` lấy kiểu của đối số
- Gọi `checkTypeEquality(argType, param->paramAttrs->type)`
- `checkTypeEquality()` so sánh bằng `compareType()` và báo lỗi nếu khác
- Vị trí:
  - + `compileArgument()`: parser.c, dòng ~573
  - + `checkTypeEquality()`: semantics.c, dòng ~145

## 6. Truyền tham biến bằng một hằng biểu diễn bởi định danh



Lỗi: 13-15:An identifier expected.

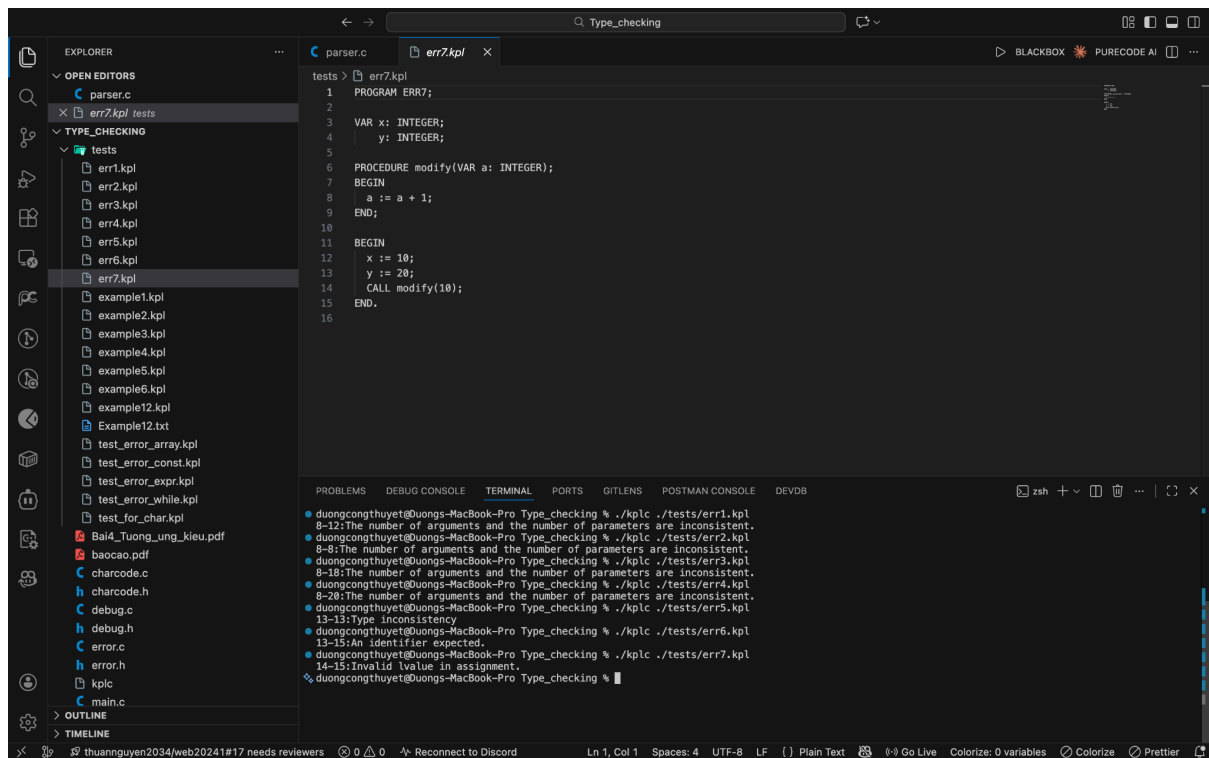
Nguyên nhân:

- Procedure modify có tham số VAR x: INTEGER (tham chiếu)
- Gọi CALL modify(MAX) với MAX là CONST (hằng số)
- Tham số VAR yêu cầu lvalue (biến) nhưng MAX là constant

Hàm phát hiện lỗi: compileLValue() -> checkDeclaredLValueIdent()

- compileArgument() gọi compileLValue() cho tham số VAR
- compileLValue() gọi checkDeclaredLValueIdent()
- checkDeclaredLValueIdent() kiểm tra obj->kind
- MAX có kind = OBJ\_CONSTANT (không phải OBJ\_VARIABLE, OBJ\_PARAMETER, OBJ\_FUNCTION)
- Điều kiện: default case báo error(ERR\_INVALID\_IDENT)
- Vị trí:
  - + compileArgument(): parser.c, dòng ~577
  - + compileLValue(): parser.c, dòng ~455
  - + checkDeclaredLValueIdent(): semantics.c, dòng ~109

7. Truyền tham biến bằng một biểu thức.



Lỗi: 14-15:Invalid lvalue in assignment.

Nguyên nhân:

- Procedure modify có tham số VAR a: INTEGER (tham chiếu)
- Gọi CALL modify(10) với 10 là số literal (TK\_NUMBER)
- Tham số VAR yêu cầu lvalue (identifier) nhưng 10 không phải identifier

Hàm phát hiện lỗi: compileArgument()

- Kiểm tra lookAhead->tokenType != TK\_IDENT
- Điều kiện: if (lookAhead->tokenType != TK\_IDENT) error(ERR\_INVALID\_LVALUE)
- Vị trí: parser.c, dòng ~576

Lưu ý: Nếu truyền biểu thức phức tạp như x+y, lỗi sẽ là "Missing ')" vì parser chỉ đọc được x rồi gặp dấu + (không phải dấu ) như expected).