Báo cáo LAB02
IT3323 Mã lớp 161269
Bài 2 : Phân tích cú pháp

Họ và tên: Dương Công Thuyết

MSSV: 20225932

I. Giải thích logic code đã implement

1. Type Declarations (Khai báo kiểu)

compileTypeDecls()

Chức năng: Xử lý nhiều khai báo kiểu liên tiếp

Hoạt động: Vòng lặp while kiểm tra xem token tiếp theo có phải là TK_IDENT không, nếu có thì gọi compileTypeDecl() để xử lý từng khai báo kiểu

Ví dụ: TYPE myArray = ARRAY [10] OF INTEGER; myChar = CHAR;

compileTypeDecl()

Chức năng: Xử lý một khai báo kiểu đơn

Hoạt động:

Đọc tên kiểu (TK_IDENT)

Đọc dấu "=" (SB_EQ)

Gọi compileType() để xử lý định nghĩa kiểu

Đọc dấu ";" (SB_SEMICOLON)

Cú pháp: identifier = Type;

2. Variable Declarations (Khai báo biến)

compileVarDecls()

Chức năng: Xử lý nhiều khai báo biến liên tiếp

Hoạt động: Tương tự compileTypeDecls(), vòng lặp while kiểm tra TK_IDENT và gọi compileVarDecl()

Ví dụ: VAR x : INTEGER; y : CHAR;

compileVarDecl()

Chức năng: Xử lý một khai báo biến đơn

Hoạt động:

Đọc tên biến (TK_IDENT)

Đọc dấu ":" (SB_COLON)

Gọi compileType() để xác định kiểu dữ liệu

Đọc dấu ";" (SB_SEMICOLON)

Cú pháp: identifier : Type;

3. Subroutine Declarations (Khai báo chương trình con)

compileSubDecls()

Chức năng: Xử lý tất cả các khai báo hàm (FUNCTION) và thủ tục (PROCEDURE)

Hoạt động: Vòng lặp while kiểm tra token là KW_FUNCTION hoặc KW_PROCEDURE, sau đó gọi hàm tương ứng

Điểm đặc biệt: Cho phép khai báo nhiều function/procedure liên tiếp

compileFuncDecl()

Chức năng: Xử lý khai báo hàm (có giá trị trả về)

Hoạt động:

Đọc từ khóa FUNCTION

Đọc tên hàm (TK_IDENT)

Nếu có "(", đọc danh sách tham số và đóng ")"

Đọc dấu ":" (SB_COLON)

Gọi compileBasicType() để xác định kiểu trả về

Đọc dấu ";" (SB_SEMICOLON)

Gọi compileBlock() để xử lý thân hàm

Đọc dấu ";" kết thúc

Cú pháp: FUNCTION name(params) : Type; Block;

compileProcDecl()

Chức năng: Xử lý khai báo thủ tục (không có giá trị trả về)

Hoạt động: Tương tự compileFuncDecl() nhưng không có phần khai báo kiểu trả về

Cú pháp: PROCEDURE name(params); Block;

4. Constants (Hằng số)

compileUnsignedConstant()

Chức năng: Xử lý hằng số không dấu

Hoạt động: Sử dụng switch-case để kiểm tra và đọc:

TK_NUMBER: số nguyên

TK_IDENT: tên hằng đã định nghĩa trước

TK_CHAR: ký tự

Nếu không phải các loại trên → báo lỗi ERR_INVALID_CONSTANT

compileConstant()

Chức năng: Xử lý hằng số có thể có dấu (+/-)

Hoạt động:

Nếu có SB_PLUS hoặc SB_MINUS → đọc dấu rồi gọi compileConstant2()

Nếu là TK_CHAR → đọc trực tiếp

Ngược lại → gọi compileConstant2()

Ví dụ: +5, -10, 'a', MAX_VALUE

compileConstant2()

Chức năng: Xử lý phần số hoặc identifier của hằng

Hoạt động: Kiểm tra và đọc TK_IDENT hoặc TK_NUMBER, báo lỗi nếu không phải

5. Basic Types (Kiểu cơ bản)

compileBasicType()

Chức năng: Xử lý các kiểu dữ liệu cơ bản

Hoạt động: Switch-case kiểm tra:

KW_INTEGER → kiểu số nguyên

KW_CHAR → kiểu ký tự
Nếu không phải → báo lỗi ERR_INVALIDBASICTYPE

6. Parameters (Tham số)

compileParams()
Chức năng: Xử lý danh sách tham số
Hoạt động: Gọi compileParam() cho tham số đầu tiên, sau đó gọi compileParams2() cho các tham số còn lại
compileParams2()
Chức năng: Xử lý các tham số bổ sung
Hoạt động: Đệ quy - nếu gặp ";" thì đọc tham số tiếp theo và gọi lại chính nó
Cú pháp: param1 : Type; param2 : Type; ...
compileParam()
Chức năng: Xử lý một tham số đơn
Hoạt động:
Nếu bắt đầu bằng TK_IDENT: tham số truyền giá trị name : Type
Nếu bắt đầu bằng KW_VAR: tham số truyền tham chiếu VAR name : Type
Ngược lại → báo lỗi ERR_INVALIDPARAM

7. Statements (Các câu lệnh)

compileAssignSt()
Chức năng: Xử lý câu lệnh gán
Hoạt động:
Đọc tên biến (TK_IDENT)
Gọi compileIndexes() để xử lý chỉ số mảng (nếu có)
Đọc toán tử gán ":=" (SB_ASSIGN)
Gọi compileExpression() để tính giá trị vế phải
Cú pháp: variable := expression hoặc array[index] := expression
compileCallSt()
Chức năng: Xử lý câu lệnh gọi thủ tục/hàm
Hoạt động:
Đọc từ khóa CALL
Đọc tên thủ tục (TK_IDENT)
Nếu có "(", đọc danh sách đối số và đóng ")"
Cú pháp: CALL procedureName hoặc CALL procedureName(args)
compileGroupSt()
Chức năng: Xử lý khối lệnh (compound statement)
Hoạt động:
Đọc BEGIN
Gọi compileStatements() để xử lý các câu lệnh bên trong
Đọc END
Cú pháp: BEGIN statements END
compileWhileSt()
Chức năng: Xử lý vòng lặp while

Hoạt động:
Đọc từ khóa WHILE
Gọi compileCondition() để xử lý điều kiện
Đọc từ khóa DO
Gọi compileStatement() để xử lý câu lệnh lặp
Cú pháp: WHILE condition DO statement
compileForSt()
Chức năng: Xử lý vòng lặp for
Hoạt động:
Đọc từ khóa FOR
Đọc biến đếm (TK_IDENT)
Đọc ":=" (SB_ASSIGN)
Gọi compileExpression() cho giá trị bắt đầu
Đọc từ khóa TO
Gọi compileExpression() cho giá trị kết thúc
Đọc từ khóa DO
Gọi compileStatement() cho thân vòng lặp
Cú pháp: FOR i := start TO end DO statement

8. Arguments (Đối số)

compileArguments()
Chức năng: Xử lý danh sách đối số khi gọi hàm/thủ tục
Hoạt động: Gọi compileExpression() cho đối số đầu tiên, sau đó compileArguments2() cho các đối số còn lại
compileArguments2()
Chức năng: Xử lý các đối số bổ sung
Hoạt động: Đệ quy - nếu gặp "," thì đọc biểu thức tiếp theo
Cú pháp: arg1, arg2, arg3, ...

9. Conditions (Điều kiện)

compileCondition()
Chức năng: Xử lý điều kiện so sánh
Hoạt động: Gọi compileExpression() cho vế trái, sau đó compileCondition2() cho toán tử và vế phải
compileCondition2()
Chức năng: Xử lý toán tử so sánh và vế phải
Hoạt động: Switch-case kiểm tra các toán tử:
SB_EQ: bằng (=)
SB_NEQ: khác (<>)
SB_LT: nhỏ hơn (<)
SB_LE: nhỏ hơn hoặc bằng (<=)
SB_GT: lớn hơn (>)
SB_GE: lớn hơn hoặc bằng (>=)
Sau đó gọi compileExpression() cho vế phải

Cú pháp: expression comparator expression

10. Expressions (Biểu thức)

compileExpression()

Chức năng: Xử lý biểu thức với dấu (+/-) ở đầu (optional)

Hoạt động:

Nếu có SB_PLUS hoặc SB_MINUS → đọc dấu rồi gọi compileExpression2()

Ngược lại → gọi trực tiếp compileExpression2()

Ví dụ: +5, -x, a + b

compileExpression2()

Chức năng: Xử lý term và các phép cộng/trừ tiếp theo

Hoạt động: Gọi compileTerm() cho term đầu tiên, sau đó compileExpression3() cho các phép toán tiếp theo

compileExpression3() (đã có sẵn)

Chức năng: Xử lý các phép cộng/trừ liên tiếp (đệ quy trái)

Hoạt động: Nếu gặp + hoặc -, đọc toán tử, gọi compileTerm(), rồi gọi lại chính nó

11. Factor (Thừa số)

compileFactor()

Chức năng: Xử lý các thừa số cơ bản trong biểu thức

Hoạt động: Switch-case kiểm tra:

TK_NUMBER: Đọc số nguyên

TK_CHAR: Đọc ký tự

TK_IDENT: Đọc identifier, sau đó:

Nếu có "(": gọi hàm với đối số → compileArguments()

Ngược lại: biến hoặc mảng → compileIndexes()

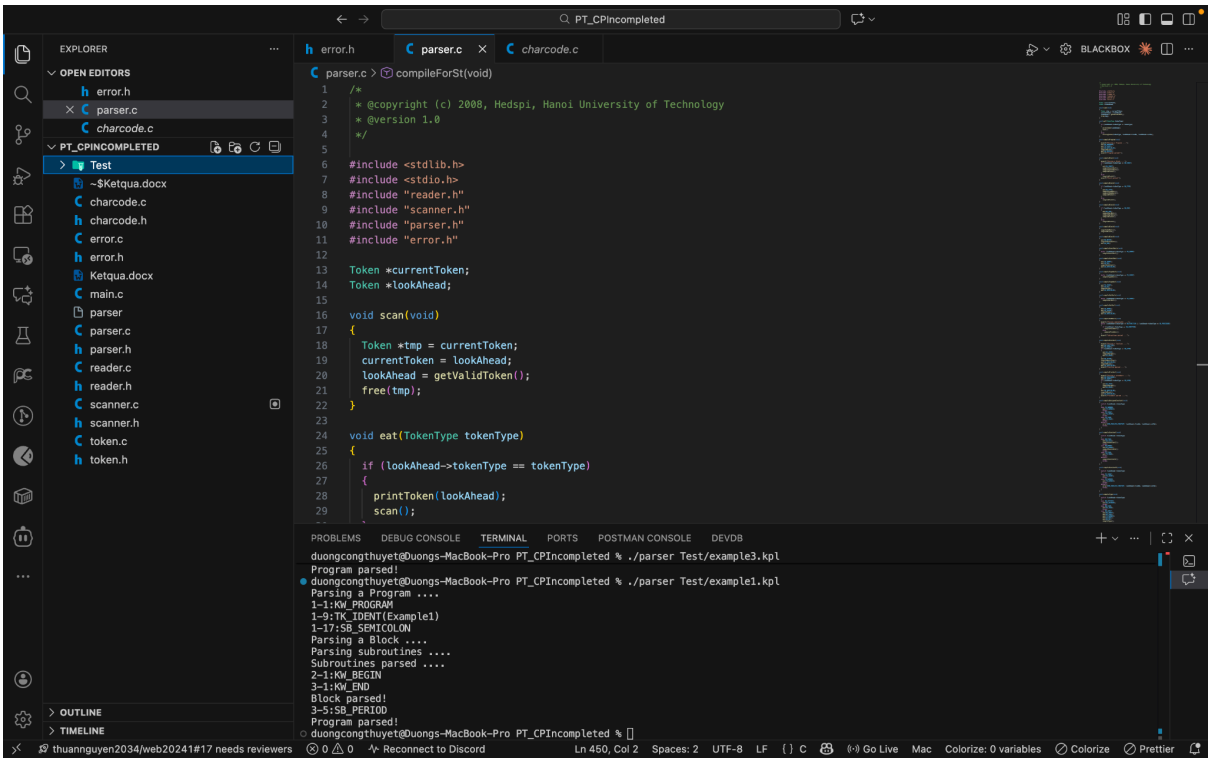
SB_LPAR: Biểu thức trong ngoặc → đọc "(", compileExpression(), đọc ")"

Ngược lại → báo lỗi ERR_INVALIDFACTOR

Ví dụ: 5, 'a', x, arr[i], func(a,b), (x+y)

II. Kết quả chạy với chương trình không lỗi

Example1:



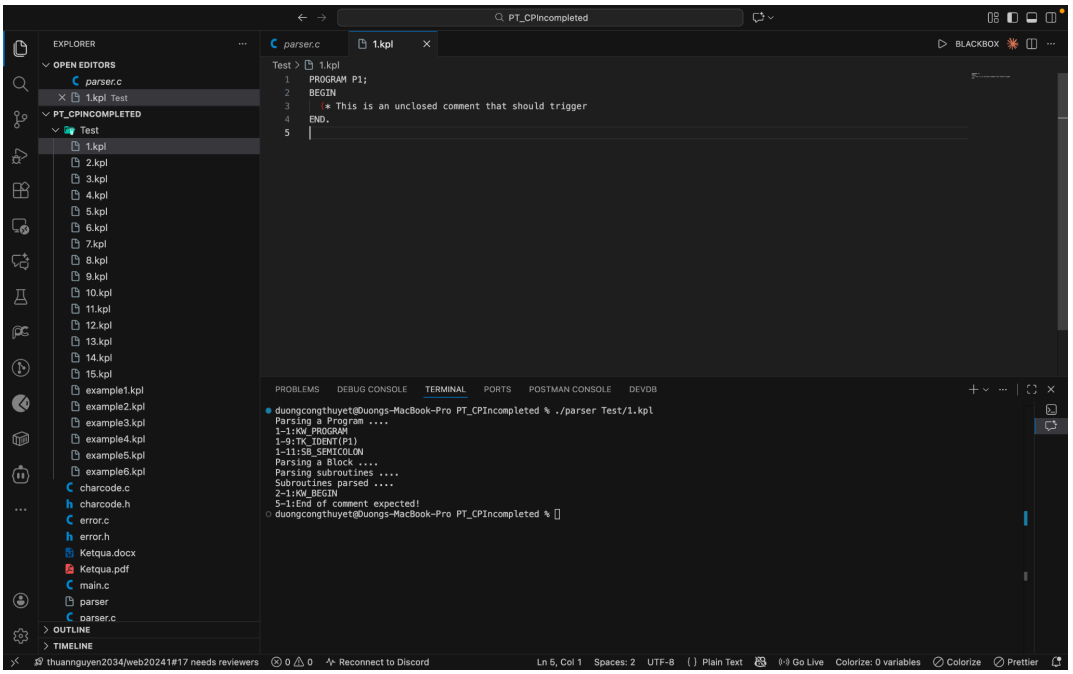
Các example còn lại được ghi trong file Ketqua.pdf

III. Các trường hợp có lỗi

1. Lỗi ERR_END_OF_COMMENT

Lỗi ERR_END_OF_COMMENT ở dòng 4: Comment bắt đầu bằng (* nhưng không có dấu đóng *)

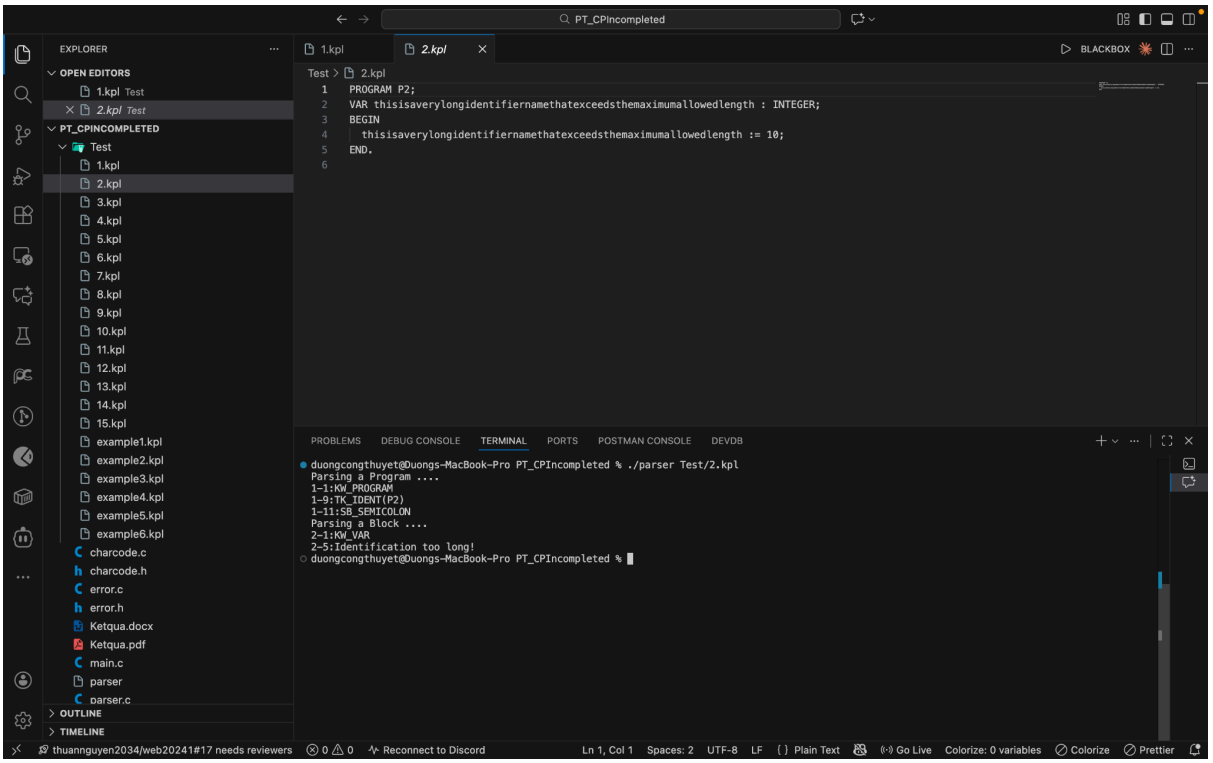
Đoạn code lỗi: " This is an unclosed comment..." - thiếu dấu *) kết thúc



2. Lỗi ERR_IDENT_TOO_LONG

Lỗi ERR_IDENT_TOO_LONG ở dòng 3: Tên biến có 63 ký tự, vượt quá giới hạn cho phép

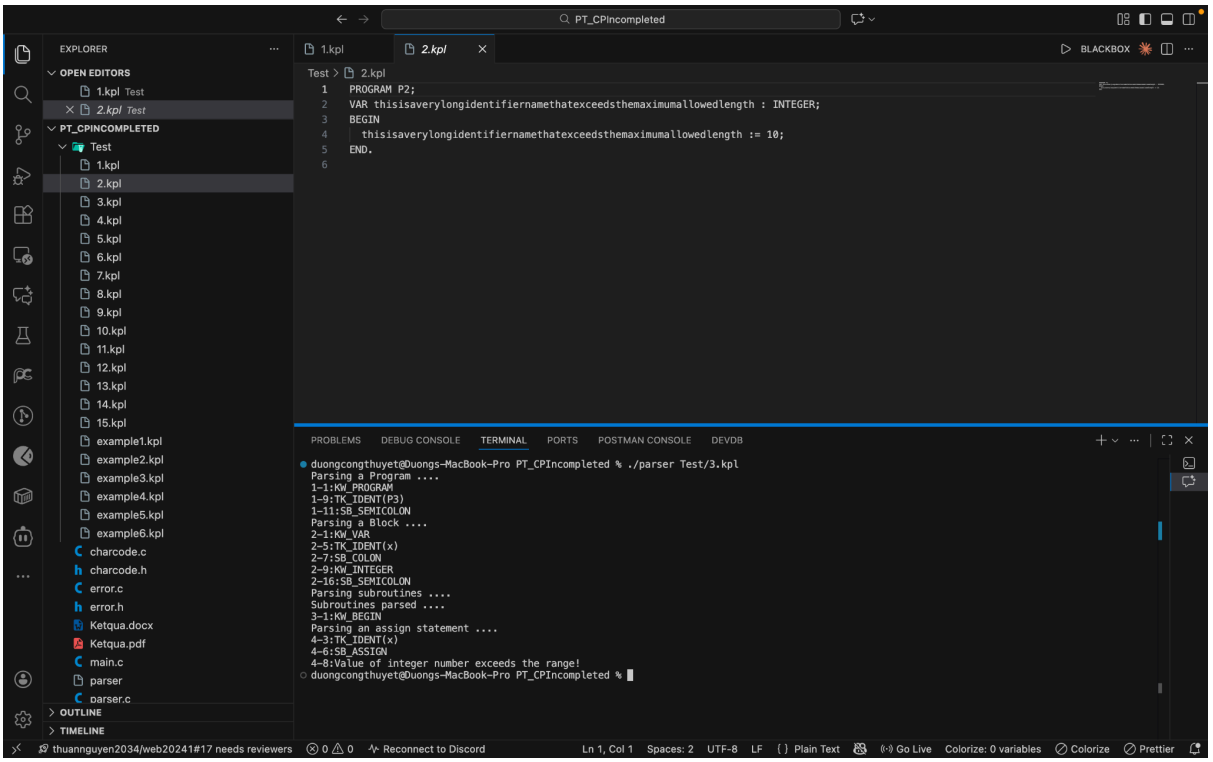
Đoạn code lỗi: "thisisaverylongidentifiernamethatexceedsthemaximumallowedlength"



3. Lỗi ERR_NUMBER_TOO_LONG

Lỗi ERR_NUMBER_TOO_LONG ở dòng 3: Số 99999999999999999999 vượt quá phạm vi INTEGER

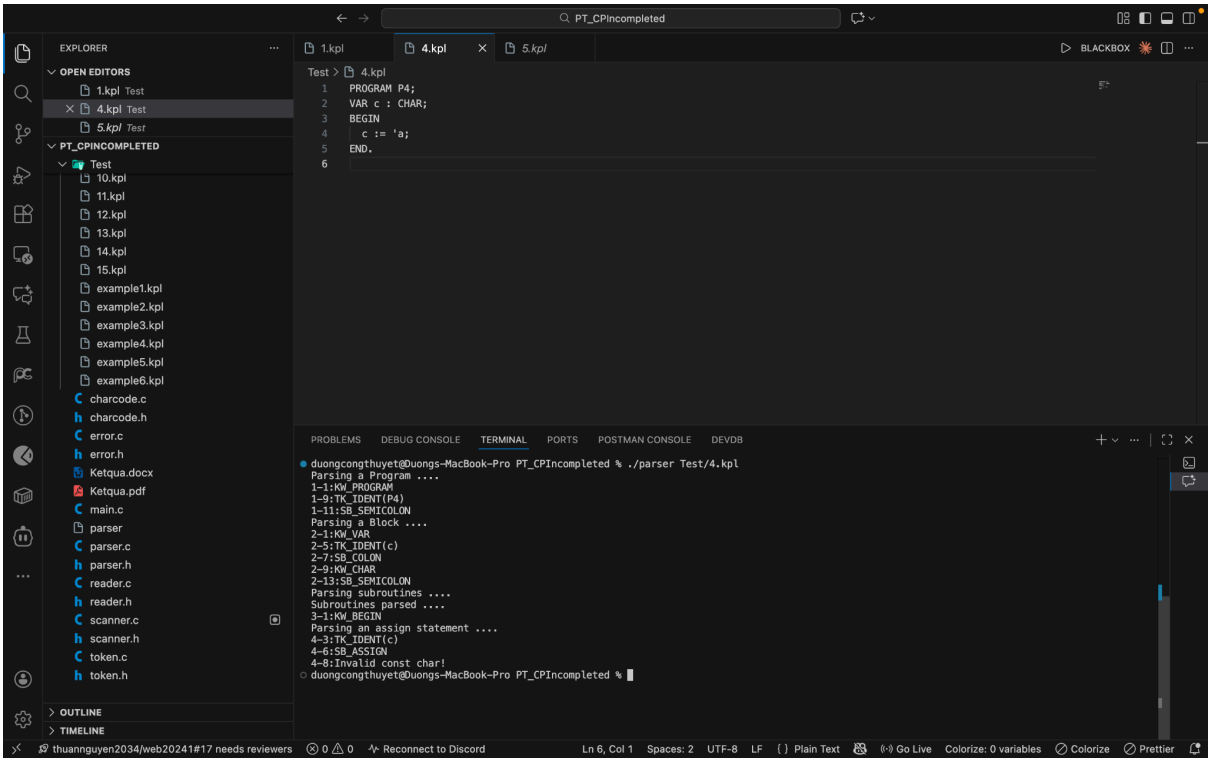
Đoạn code lỗi: "max = 99999999999999999999" - số có 20 chữ số



4. Lỗi ERR_INVALIDCHARCONSTANT

Lỗi ERR_INVALIDCHARCONSTANT ở dòng 5: Hằng ký tự 'a; thiếu dấu ngoặc đơn đóng

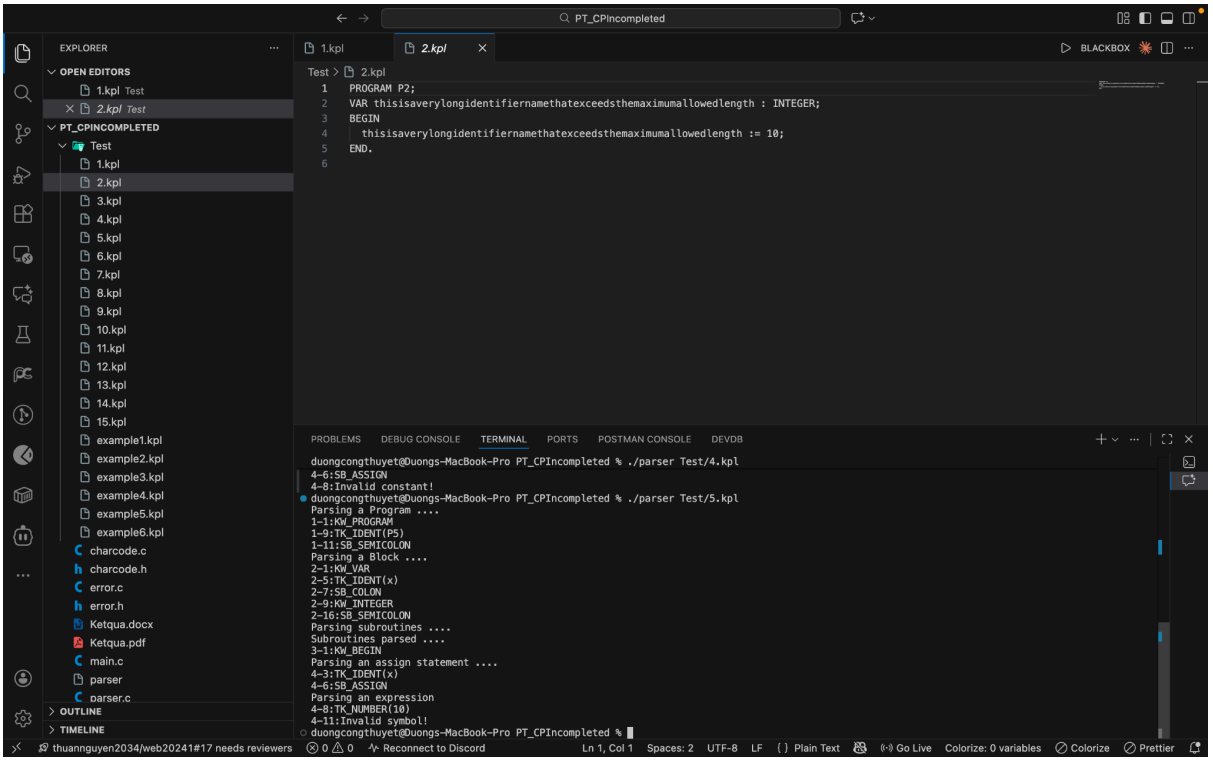
Đoạn code lỗi: "c := 'a;" - phải là "c := 'a';"



5. Lỗi ERR_INVALID_SYMBOL

Lỗi ERR_INVALID_SYMBOL ở dòng 4: Ký hiệu @ không hợp lệ trong KPL

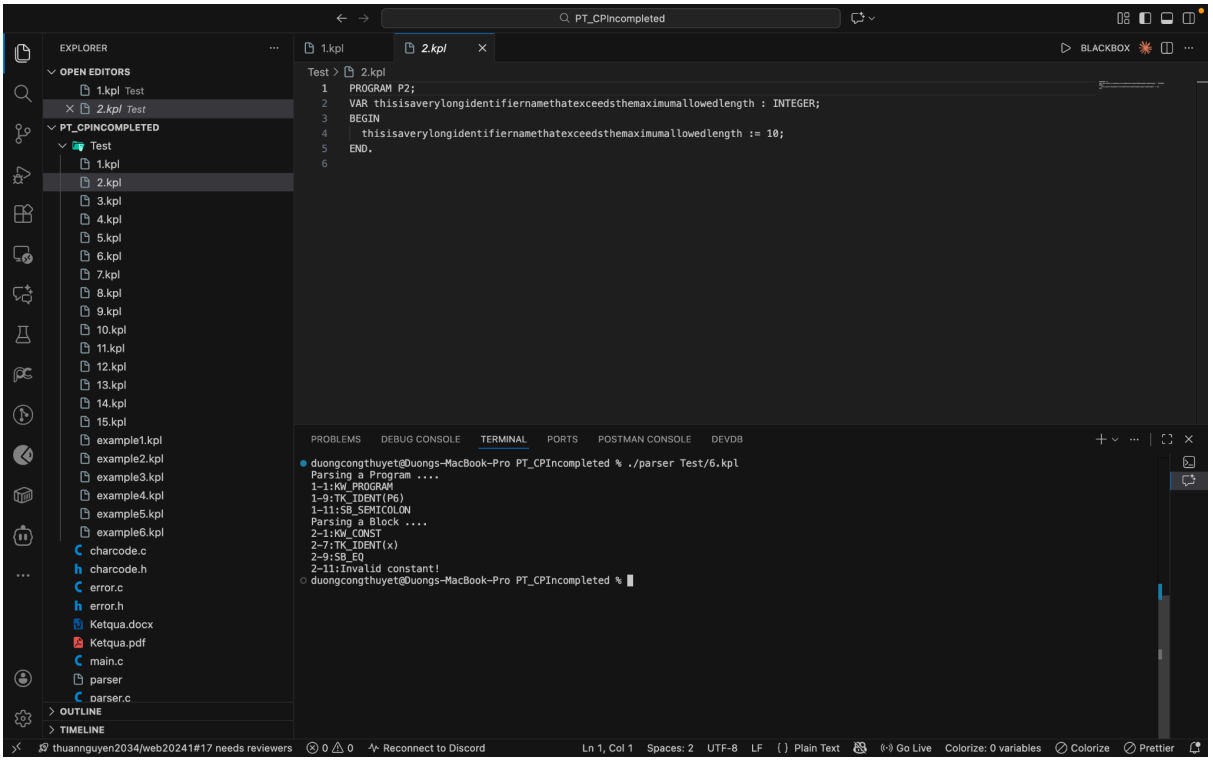
Đoạn code lỗi: "x := 5 @ 3;" - ký tự @ không thuộc bộ ký hiệu của ngôn ngữ



6. Lỗi ERR_INVALID_CONSTANT

Lỗi ERR_INVALID_CONSTANT ở dòng 3: Khai báo CONST thiếu giá trị sau dấu =

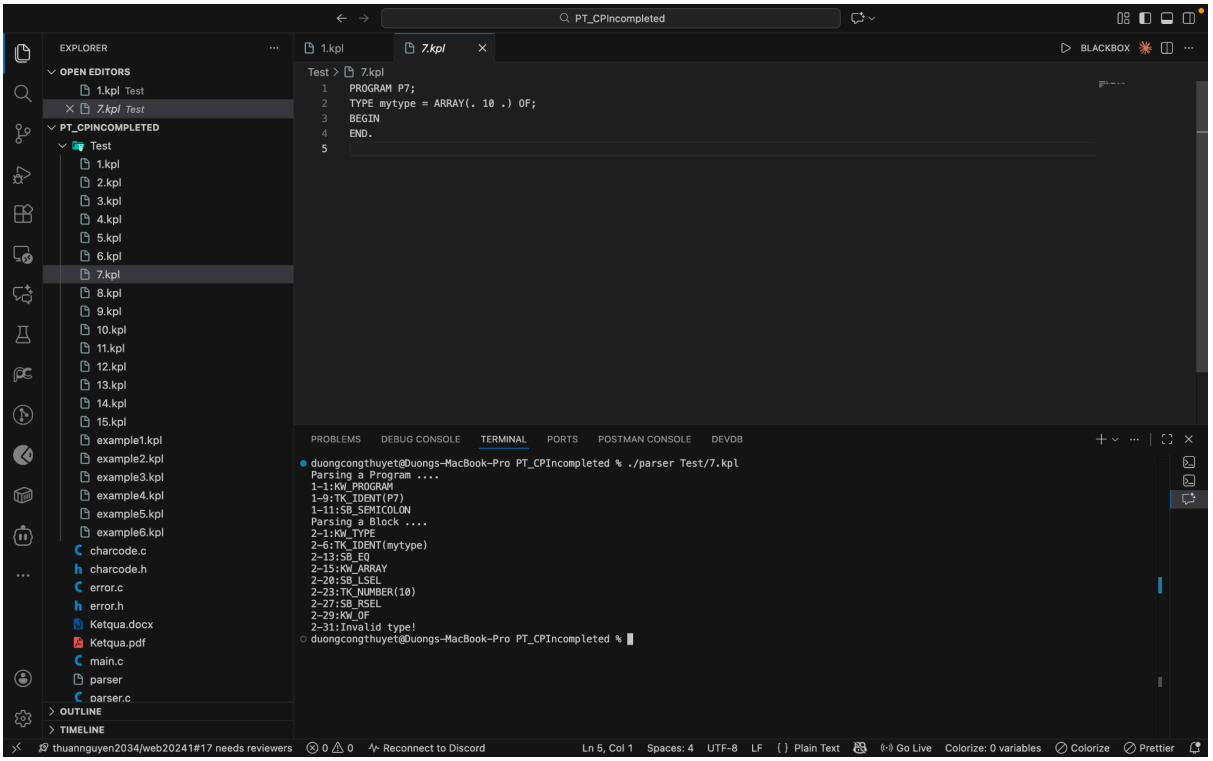
Đoạn code lỗi: "CONST x = ;" - thiếu giá trị hằng số giữa = và ;



7. Lỗi ERR_INVALIDTYPE

Lỗi ERR_INVALIDTYPE ở dòng 3: Khai báo ARRAY sai cú pháp, thiếu [size] OF

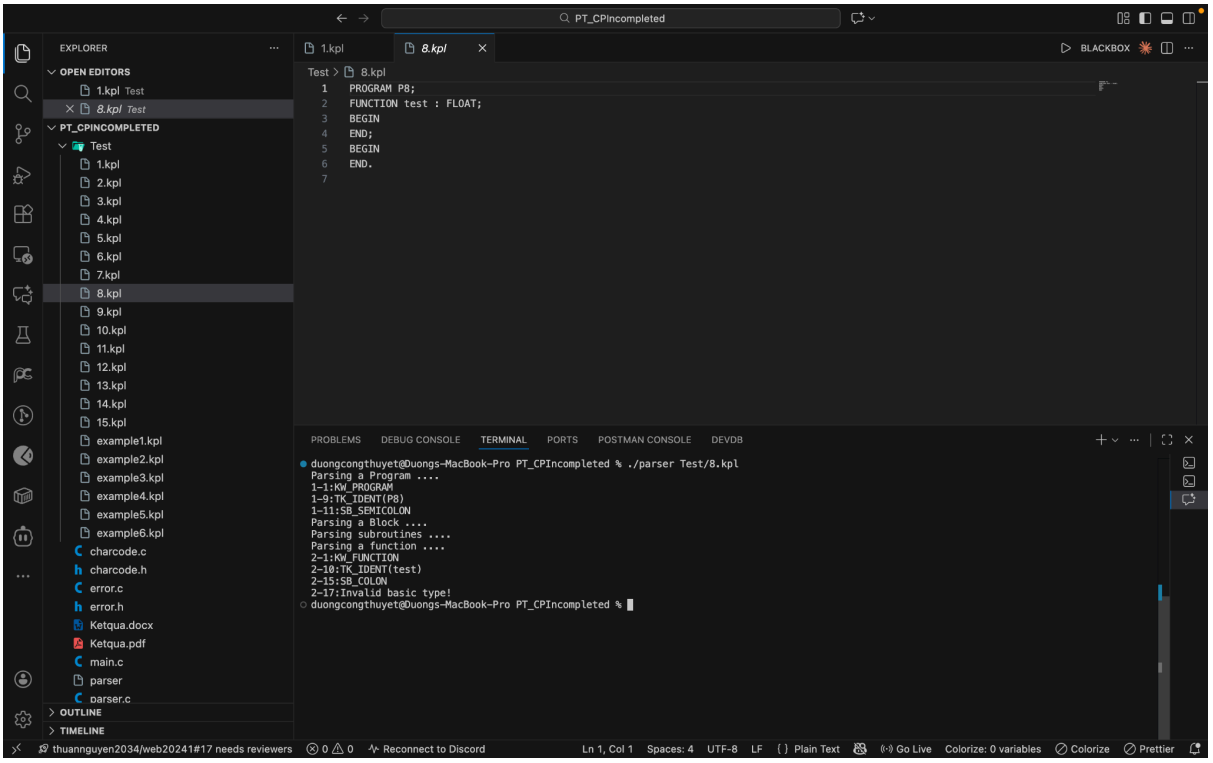
Đoạn code lỗi: "arr = ARRAY INTEGER" - phải là "arr = ARRAY [10] OF INTEGER"



8. Lỗi ERR_INVALIDBASICTYPE

Lỗi ERR_INVALIDBASICTYPE ở dòng 3: FLOAT không phải kiểu cơ bản hợp lệ

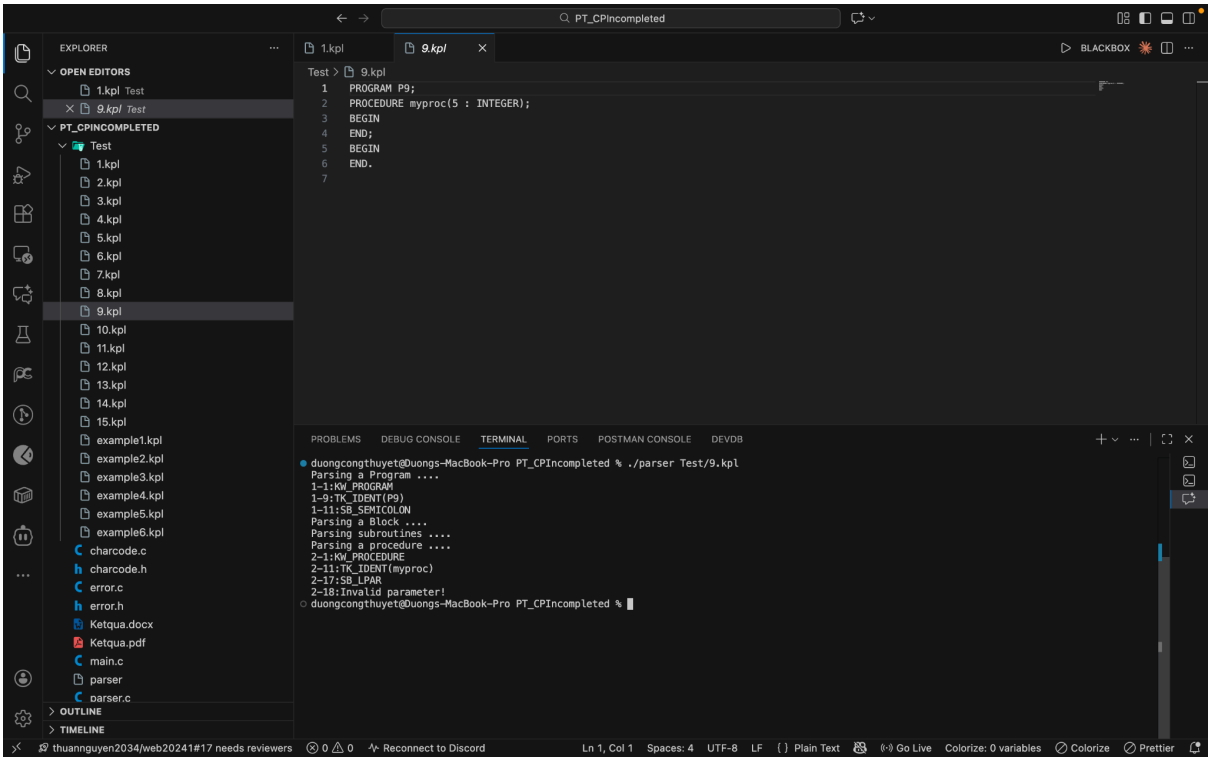
Đoạn code lỗi: "FUNCTION test : FLOAT;" - chỉ có INTEGER và CHAR là hợp lệ



9. Lỗi ERR_INVALIDPARAM

Lỗi ERR_INVALIDPARAM ở dòng 3: Tham số thiếu dấu : và kiểu dữ liệu

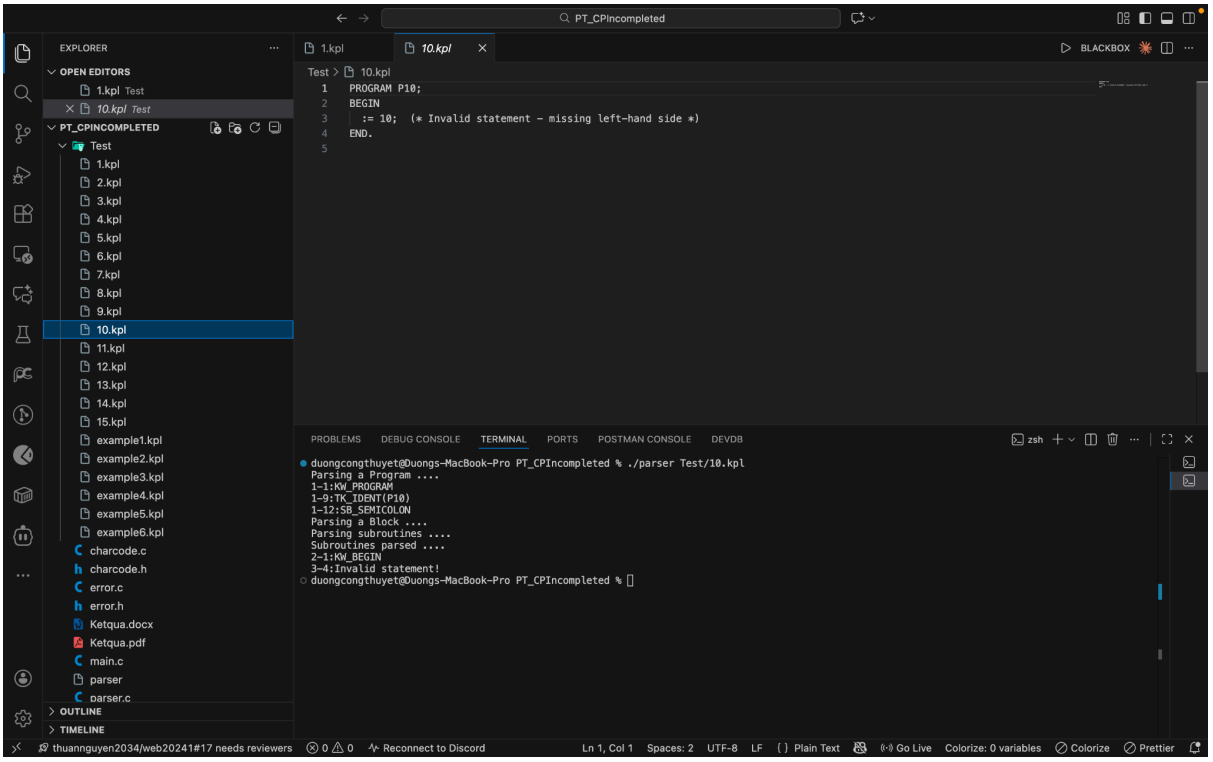
Đoạn code lỗi: "test(x y)" - phải là "test(x : INTEGER; y : INTEGER)"



10. Lỗi ERR_INVALIDSTATEMENT

Lỗi ERR_INVALIDSTATEMENT ở dòng 4: RETURN không phải là câu lệnh hợp lệ

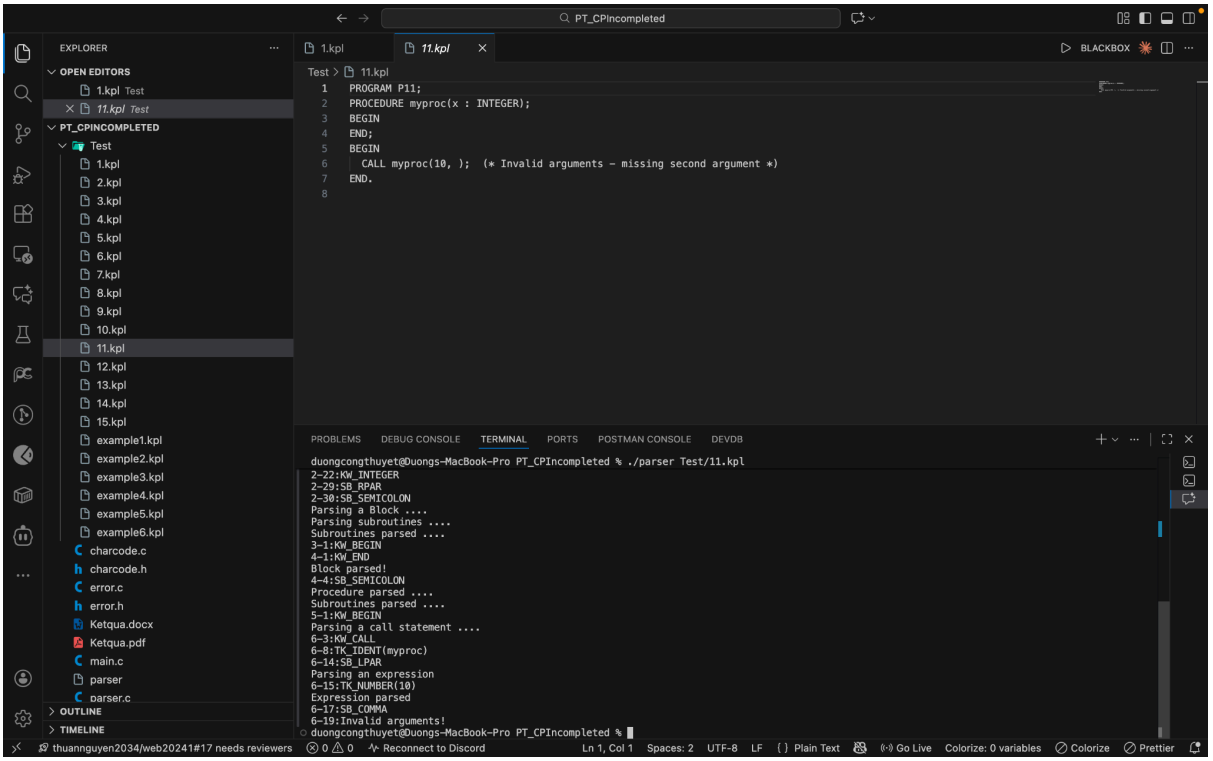
Đoạn code lỗi: "RETURN 5;" - KPL không có câu lệnh RETURN



11. Lỗi ERR_INVALIDARGUMENTS

Lỗi ERR_INVALIDARGUMENTS ở dòng 8: Danh sách đối số thiếu biểu thức sau dấu phẩy

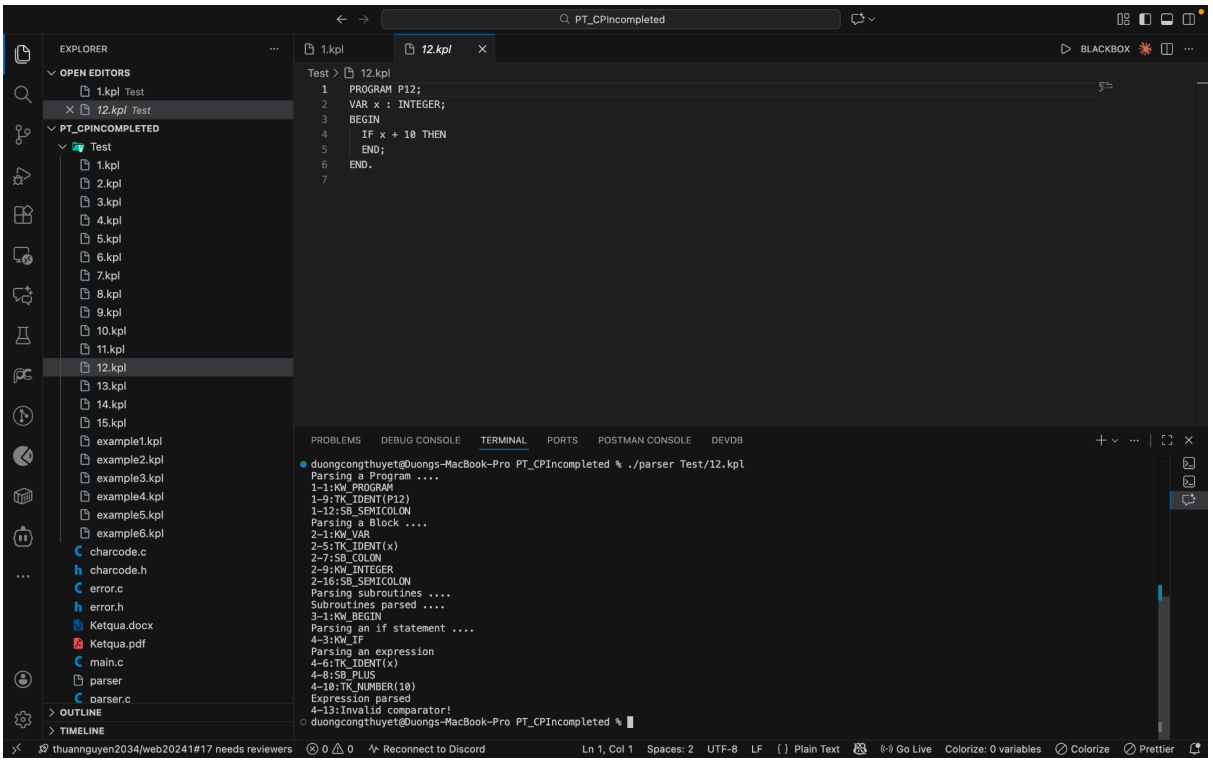
Đoạn code lỗi: "CALL test(5,);" - thiếu đối số thứ 2 sau dấu phẩy



12. Lỗi ERR_INVALIDCOMPARATOR

Lỗi ERR_INVALIDCOMPARATOR ở dòng 5: Toán tử >> không phải toán tử so sánh hợp lệ

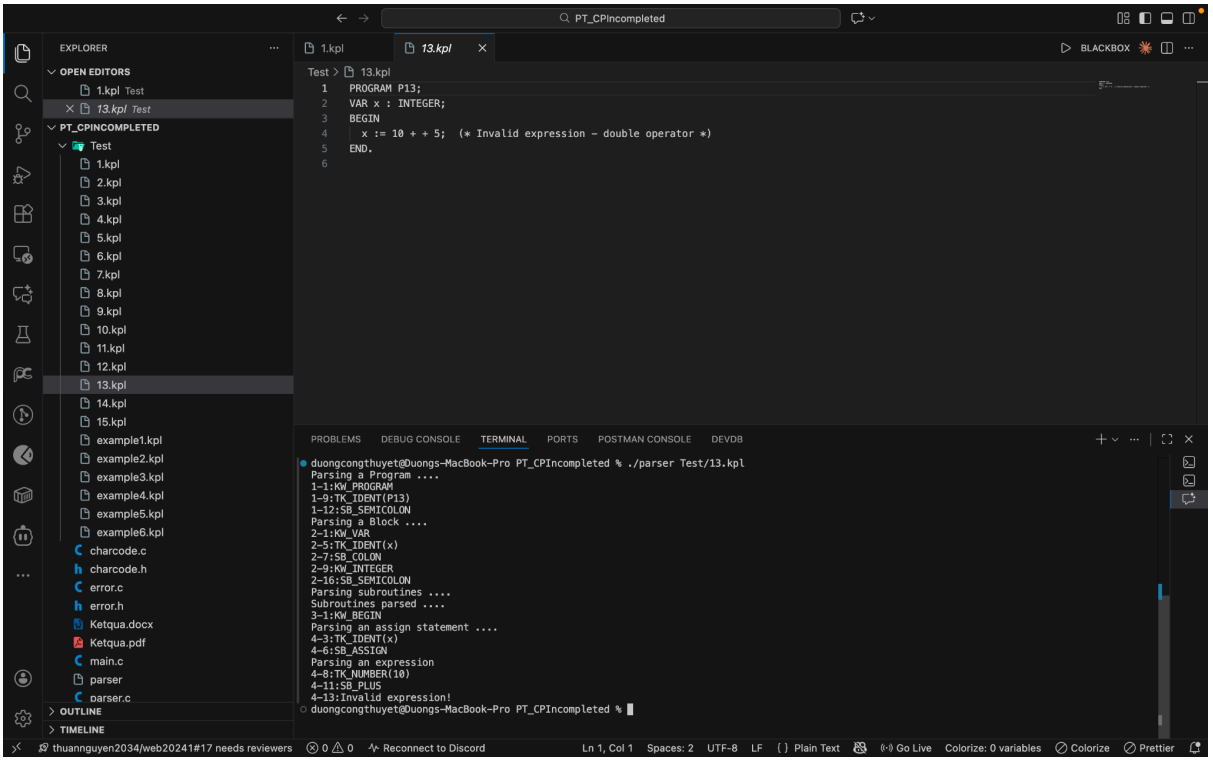
Đoạn code lỗi: "IF x >> 5" - chỉ có =, <>, <, <=, >, >= là hợp lệ



13. Lỗi ERR_INVALIDEXPRESSION

Lỗi ERR_INVALIDEXPRESSION ở dòng 5: Biểu thức thiếu toán hạng sau dấu +

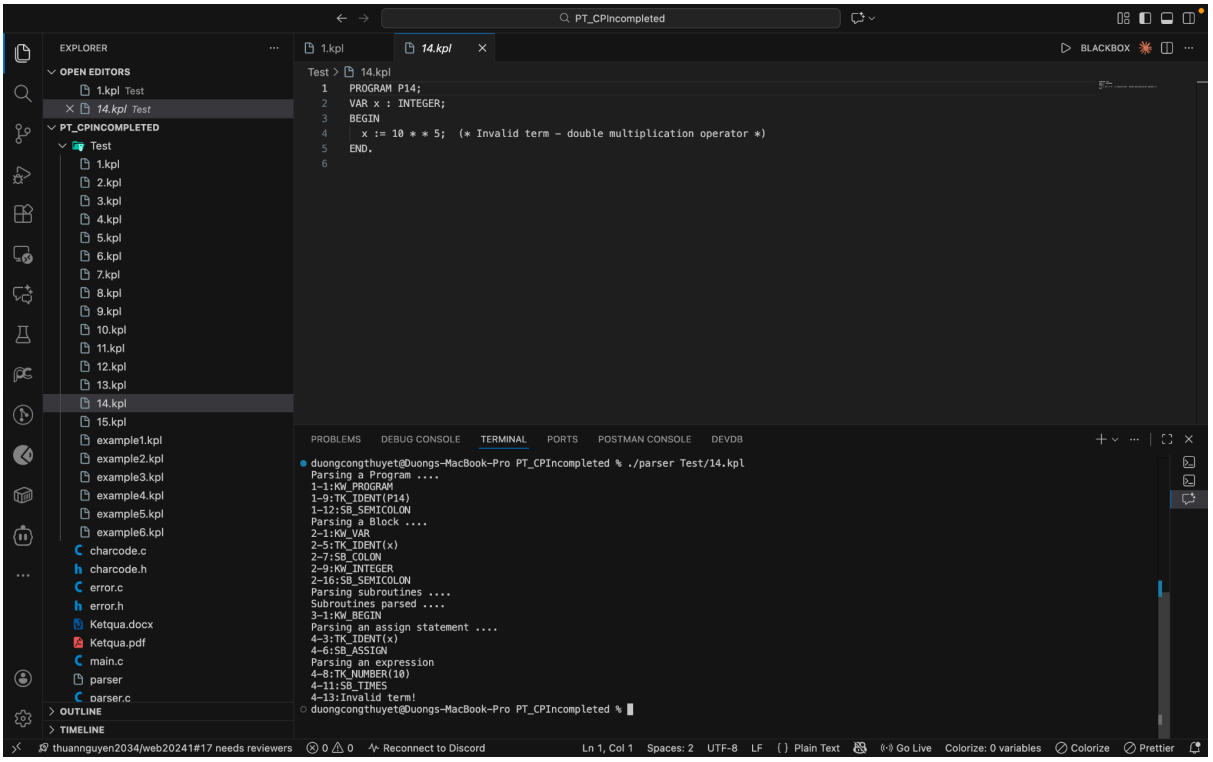
Đoạn code lỗi: "x := 5 + ;" - thiếu term/factor sau toán tử cộng



14. Lỗi ERR_INVALIDTERM

Lỗi ERR_INVALIDTERM ở dòng 5: Term thiếu factor sau toán tử *

Đoạn code lỗi: "x := 5 * ;" - thiếu factor sau toán tử nhân



15. Lỗi ERR_INVALIDFACTOR

Lỗi ERR_INVALIDFACTOR ở dòng 5: BEGIN không phải là factor hợp lệ

Đoạn code lỗi: "x := BEGIN;" - factor chỉ có thể là số, ký tự, biến, hàm, hoặc (expression)