BÁO CÁO THỰC HÀNH XÂY DỰNG CHƯƠNG TRÌNH DỊCH

Bài 3: Tạo bảng ký hiệu

Họ và tên: Trịnh Hữu An

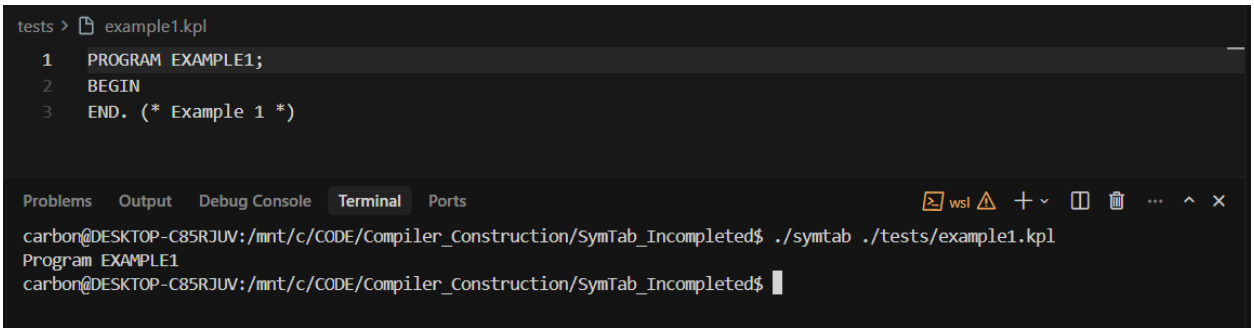
MSSV: 20225593

Mã lớp: 161629

Mục lục:

I. Kết quả thực hiện với các ví dụ:

1. Kết quả với example1.kpl



Giải thích:

1. Chương trình symtab đã parse thành công file KPL:

- Nhận diện từ khóa PROGRAM và tên chương trình EXAMPLE1
- Xử lý khối BEGIN...END (rỗng, không có khai báo)
- Kết thúc bằng dấu chấm .

2. Kết quả in ra:

- Program EXAMPLE1: tên chương trình được tạo trong symbol table
- Không có dòng nào khác vì khối chương trình rỗng (không có constant, variable, type, function, procedure)

3. Kết luận: Chương trình hợp lệ nhưng không có khai báo nào, nên chỉ in tên chương trình.

2. Kết quả với example2.kpl

tests > example2.kpl

```
1   PROGRAM EXAMPLE2; (* Factorial *)
2
3   VAR N : INTEGER;
4
5   FUNCTION F(N : INTEGER) : INTEGER;
6     BEGIN
7       IF N = 0 THEN F := 1 ELSE F := N * F (N - 1);
8     END;
9
10  BEGIN
11    FOR N := 1 TO 7 DO
12      BEGIN
13        CALL WRITELN;
14        CALL WRITEI(F(n));
15      END;
16  END. (* Factorial *)
```

Problems Output Debug Console Terminal Ports wsl

```
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$ ./symtab ./tests/example2.kpl
Program EXAMPLE2
    Var N : Int
    Function F : Int
        Param N : Int

carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$
```

Giải thích:

Khi parser xử lý file example2.kpl, symbol table được xây dựng theo cấu trúc phân cấp scope:

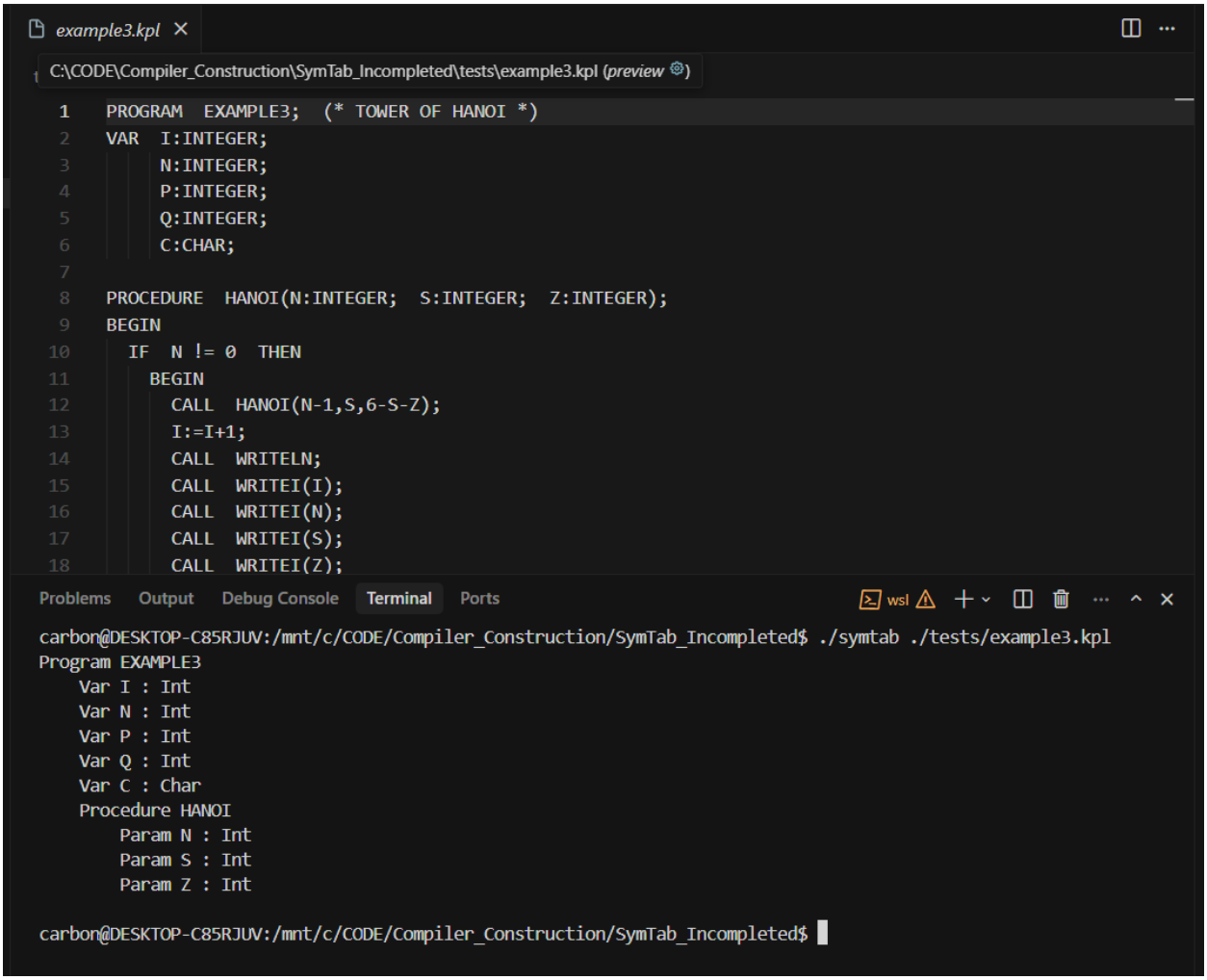
1. Program EXAMPLE2: Tên chương trình được tạo trong symbol table.
2. Var N : Int (indent 4 spaces): Biến N kiểu INTEGER được khai báo ở dòng 3 (VAR N : INTEGER;), thuộc scope của chương trình. Indent 4 spaces cho biết đây là object trong scope chương trình.
3. Function F : Int (indent 4 spaces): Hàm F trả về kiểu INTEGER được khai báo ở dòng 5 (FUNCTION F(N : INTEGER) : INTEGER;), thuộc scope chương trình. Indent 4 spaces cho biết đây là object trong scope chương trình.
4. Param N : Int (indent 8 spaces): Tham số N kiểu INTEGER của hàm F, được khai báo trong danh sách tham số của hàm (dòng 5). Indent 8 spaces cho biết đây là object trong scope của hàm F (scope con).

Nhận thấy: Có hai object cùng tên "N" nhưng không xung đột vì thuộc hai scope khác nhau:

- Var N: biến toàn cục, thuộc scope chương trình
- Param N: tham số của hàm F, thuộc scope hàm F

Khi hàm F được gọi, tham số N sẽ được ưu tiên tìm kiếm trước biến toàn cục N do quy tắc scope resolution (tìm từ scope trong ra ngoài).

3. Kết quả với example3.kpl



Giải thích:

Khi parser xử lý file example3.kpl, symbol table được xây dựng như sau:

1. Program EXAMPLE3: Tên chương trình được tạo trong symbol table.
2. Các biến toàn cục (indent 4 spaces) — được khai báo ở dòng 2-6:

- Var I : Int: Biến I kiểu INTEGER
- Var N : Int: Biến N kiểu INTEGER
- Var P : Int: Biến P kiểu INTEGER
- Var Q : Int: Biến Q kiểu INTEGER
- Var C : Char: Biến C kiểu CHAR

Tất cả thuộc scope chương trình, nên có indent 4 spaces.

3. Procedure HANOI (indent 4 spaces): Procedure HANOI được khai báo ở dòng 8 (PROCEDURE HANOI(N:INTEGER; S:INTEGER; Z:INTEGER);), thuộc scope chương trình.
4. Các tham số của procedure HANOI (indent 8 spaces) — được khai báo trong danh sách tham số (dòng 8):

- Param N : Int: Tham số N kiểu INTEGER
- Param S : Int: Tham số S kiểu INTEGER
- Param Z : Int: Tham số Z kiểu INTEGER

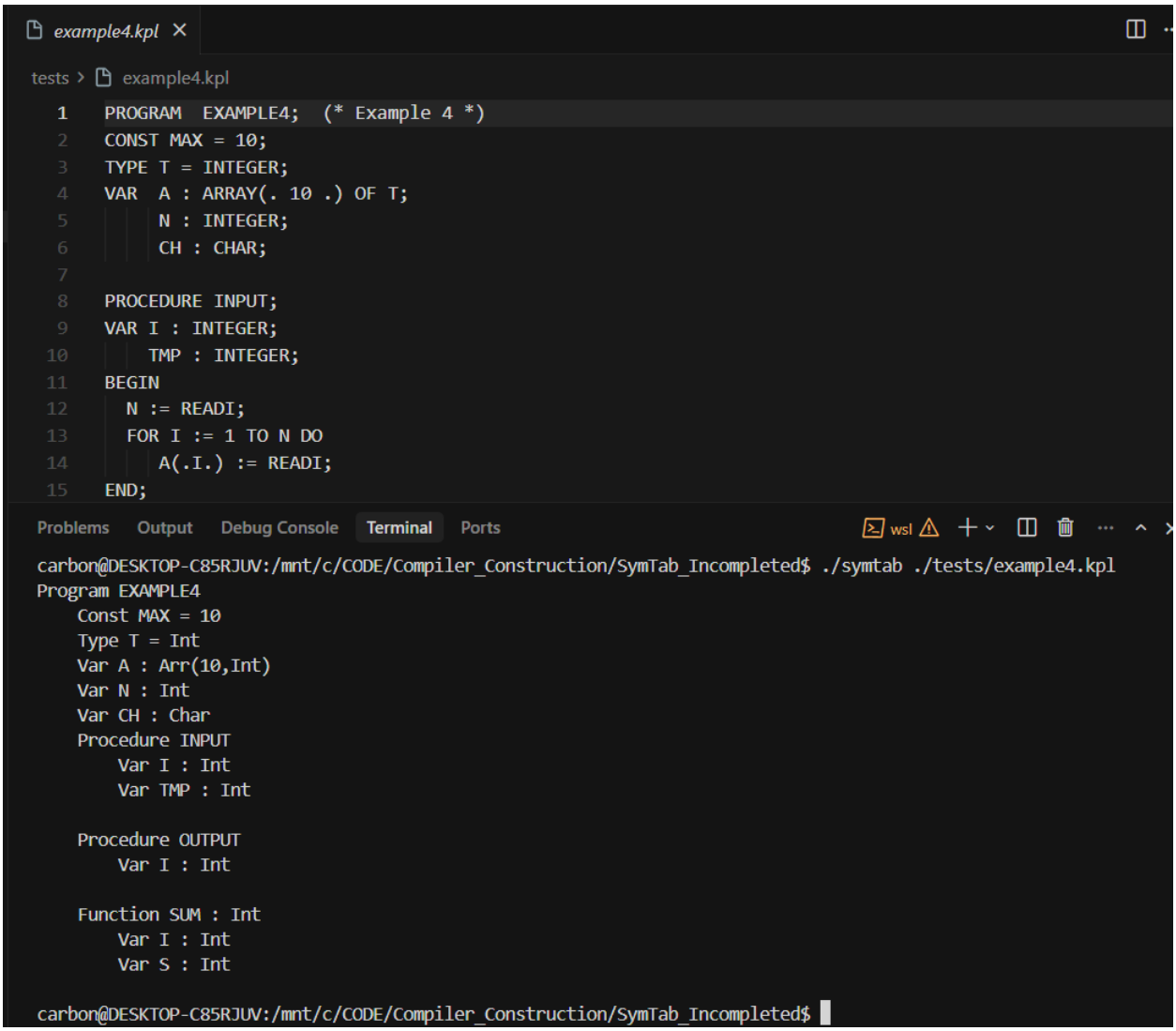
Tất cả thuộc scope của procedure HANOI, nên có indent 8 spaces.

Nhận thấy: Có hai object cùng tên "N" nhưng không xung đột:

- Var N: biến toàn cục, thuộc scope chương trình
- Param N: tham số của procedure HANOI, thuộc scope của procedure

Khi procedure HANOI được gọi, tham số N sẽ được ưu tiên tìm kiếm trước biến toàn cục N do quy tắc scope resolution (tìm từ scope trong ra ngoài).

4. Kết quả với example4.kpl



Giải thích:

Khi parser xử lý file example4.kpl, symbol table được xây dựng như sau:

1. Program EXAMPLE4: Tên chương trình được tạo trong symbol table.
2. Các khai báo toàn cục (indent 4 spaces) — được khai báo ở dòng 2-6:

- Const MAX = 10: Hằng số MAX có giá trị 10 (dòng 2: CONST MAX = 10;)
- Type T = Int: Kiểu T được định nghĩa là INTEGER (dòng 3: TYPE T = INTEGER;)
- Var A : Arr(10,Int): Mảng A có 10 phần tử kiểu INTEGER (dòng 4: VAR A : ARRAY(. 10 .) OF T; với T = INTEGER)
- Var N : Int: Biến N kiểu INTEGER (dòng 5)
- Var CH : Char: Biến CH kiểu CHAR (dòng 6)

Tất cả thuộc scope chương trình, nên có indent 4 spaces.

3. Procedure INPUT (indent 4 spaces): Procedure INPUT được khai báo ở dòng 8, thuộc scope chương trình.

Các biến cục bộ của INPUT (indent 8 spaces):

- Var I : Int: Biến I kiểu INTEGER (dòng 9)
- Var TMP : Int: Biến TMP kiểu INTEGER (dòng 10)

Thuộc scope của procedure INPUT, nên có indent 8 spaces.

4. Procedure OUTPUT (indent 4 spaces): Procedure OUTPUT được khai báo ở dòng 17, thuộc scope chương trình.

Biến cục bộ của OUTPUT (indent 8 spaces):

- Var I : Int: Biến I kiểu INTEGER (dòng 18)

Thuộc scope của procedure OUTPUT, nên có indent 8 spaces.

5. Function SUM : Int (indent 4 spaces): Hàm SUM trả về kiểu INTEGER được khai báo ở dòng 27, thuộc scope chương trình.

Các biến cục bộ của SUM (indent 8 spaces):

- Var I : Int: Biến I kiểu INTEGER (dòng 28)
- Var S : Int: Biến S kiểu INTEGER (dòng 29)

Thuộc scope của hàm SUM, nên có indent 8 spaces.

Nhận thấy: Có nhiều biến cùng tên "I" nhưng không xung đột vì thuộc các scope khác nhau:

- Không có biến I ở scope chương trình
- Mỗi procedure/function có biến I riêng trong scope của nó

5. Kết quả với example5.kpl

Giải thích:

Khi parser xử lý file example5.kpl, symbol table được xây dựng như sau:

1. Program EXAMPLE5: Tên chương trình được tạo trong symbol table.
2. Các khai báo toàn cục (indent 4 spaces) — được khai báo ở dòng 2-3:

- Const C = 1: Hằng số C có giá trị 1 (dòng 2: CONST C = 1;)
- Type T = Char: Kiểu T được định nghĩa là CHAR (dòng 3: TYPE T = CHAR;)

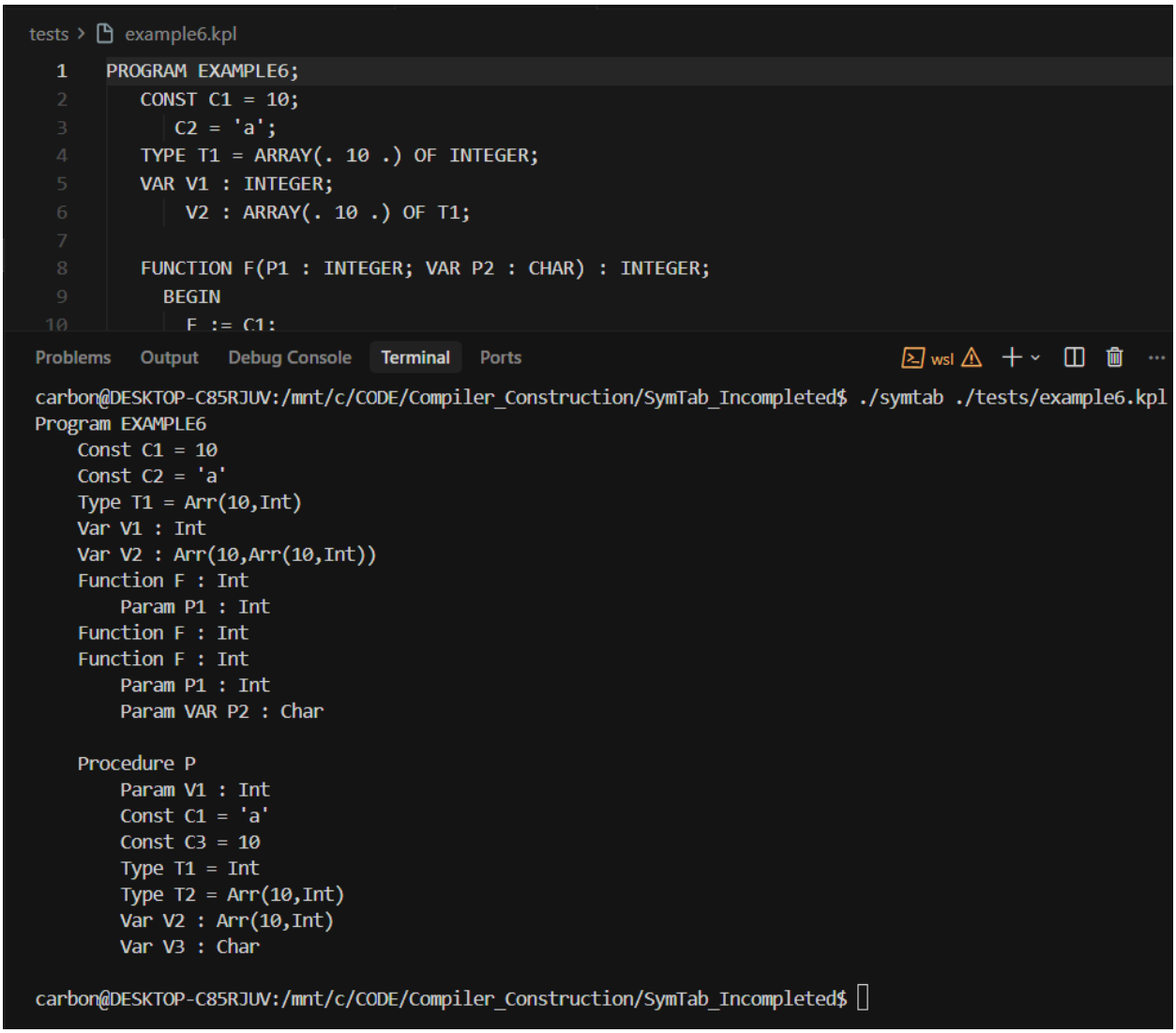
Tất cả thuộc scope chương trình, nên có indent 4 spaces.

3. Function F : Char (indent 4 spaces): Hàm F trả về kiểu CHAR được khai báo ở dòng 4 (FUNCTION F(I : INTEGER):CHAR;), thuộc scope chương trình.
4. Các khai báo trong scope của hàm F (indent 8 spaces):

- Param I : Int: Tham số I kiểu INTEGER của hàm F (dòng 4)
- Const B = 1: Hằng số B có giá trị 1 (dòng 5: CONST B = C;). Giá trị được tính từ constant C ở scope ngoài (C = 1), nên hiển thị là 1
- Type A = Arr(5,Char): Kiểu A là mảng 5 phần tử kiểu CHAR (dòng 6: TYPE A = ARRAY(.5.) OF T; với T = CHAR)

Tất cả thuộc scope của hàm F, nên có indent 8 spaces.

6. Kết quả với example6.kpl



Giải thích:

Khi parser xử lý file example6.kpl, symbol table được xây dựng như sau:

1. Program EXAMPLE6: Tên chương trình được tạo trong symbol table.
2. Các khai báo toàn cục (indent 4 spaces) — được khai báo ở dòng 2-6:

- Const C1 = 10: Hằng số C1 có giá trị 10 (dòng 2)
- Const C2 = 'a': Hằng số C2 có giá trị 'a' (dòng 3)
- Type T1 = Arr(10,Int): Kiểu T1 là mảng 10 phần tử kiểu INTEGER (dòng 4: TYPE T1 = ARRAY(. 10 .) OF INTEGER;)
- Var V1 : Int: Biến V1 kiểu INTEGER (dòng 5)

- Var V2 : Arr(10,Arr(10,Int)): Biến V2 là mảng 10 phần tử, mỗi phần tử là mảng 10 phần tử kiểu INTEGER (dòng 6: VAR V2 : ARRAY(. 10 .) OF T1; với T1 = Arr(10,Int))

Tất cả thuộc scope chương trình, nên có indent 4 spaces.

3. Function F : Int (indent 4 spaces): Hàm F trả về kiểu INTEGER được khai báo ở dòng 8, thuộc scope chương trình.

Các tham số của hàm F (indent 8 spaces):

- Param P1 : Int: Tham số P1 kiểu INTEGER (tham số truyền theo giá trị)
- Param VAR P2 : Char: Tham số P2 kiểu CHAR (tham số truyền theo tham chiếu, có từ khóa VAR)

Thuộc scope của hàm F, nên có indent 8 spaces.

4. Procedure P (indent 4 spaces): Procedure P được khai báo ở dòng 13, thuộc scope chương trình.

Các khai báo trong scope của procedure P (indent 8 spaces):

- Param V1 : Int: Tham số V1 kiểu INTEGER của procedure P (dòng 13)
- Const C1 = 'a': Hằng số C1 có giá trị 'a' (dòng 14). Lưu ý: đây là constant riêng trong scope của P, khác với C1 = 10 ở scope chương trình
- Const C3 = 10: Hằng số C3 có giá trị 10 (dòng 15)
- Type T1 = Int: Kiểu T1 được định nghĩa là INTEGER (dòng 16). Lưu ý: đây là type riêng trong scope của P, khác với T1 = Arr(10,Int) ở scope chương trình
- Type T2 = Arr(10,Int): Kiểu T2 là mảng 10 phần tử kiểu INTEGER (dòng 17: TYPE T2 = ARRAY(. 10 .) OF T1; với T1 = Int trong scope này)
- Var V2 : Arr(10,Int): Biến V2 là mảng 10 phần tử kiểu INTEGER (dòng 18: VAR V2 : T2; với T2 = Arr(10,Int))
- Var V3 : Char: Biến V3 kiểu CHAR (dòng 19)

Tất cả thuộc scope của procedure P, nên có indent 8 spaces.

Nhận thấy:

- Có nhiều object cùng tên nhưng không xung đột vì thuộc các scope khác nhau:
- Const C1 = 10 (scope chương trình) và Const C1 = 'a' (scope procedure P)
- Type T1 = Arr(10,Int) (scope chương trình) và Type T1 = Int (scope procedure P)
- Var V1 : Int (scope chương trình) và Param V1 : Int (scope procedure P)
- Var V2 : Arr(10,Arr(10,Int)) (scope chương trình) và Var V2 : Arr(10,Int) (scope procedure P)

Khi procedure P được thực thi, các object trong scope của nó sẽ được ưu tiên tìm kiếm trước các object cùng tên ở scope ngoài do quy tắc scope resolution (tìm từ scope trong ra ngoài).

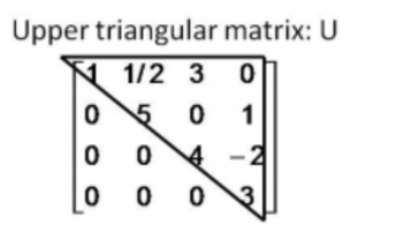
II. Thực hiện với chương trình KPL ở phần lý thuyết:

1. Bài tập 1

XÂY DỰNG CHƯƠNG TRÌNH DỊCH

Bài 1. Một ma trận vuông là ma trận tam giác trên nếu mọi phần tử nằm dưới đường chéo chính là bằng 0. Viết chương trình trên ngôn ngữ KPL để nhập một ma trận vuông kích thước nxn, n nhập từ bàn phím. In ra 1 nếu ma trận là tam giác trên , 0 nếu ngược lại.

Ví dụ một ma trận tam giác trên:



Chương trình:

assignment1.kpl

tests > assignment1.kpl

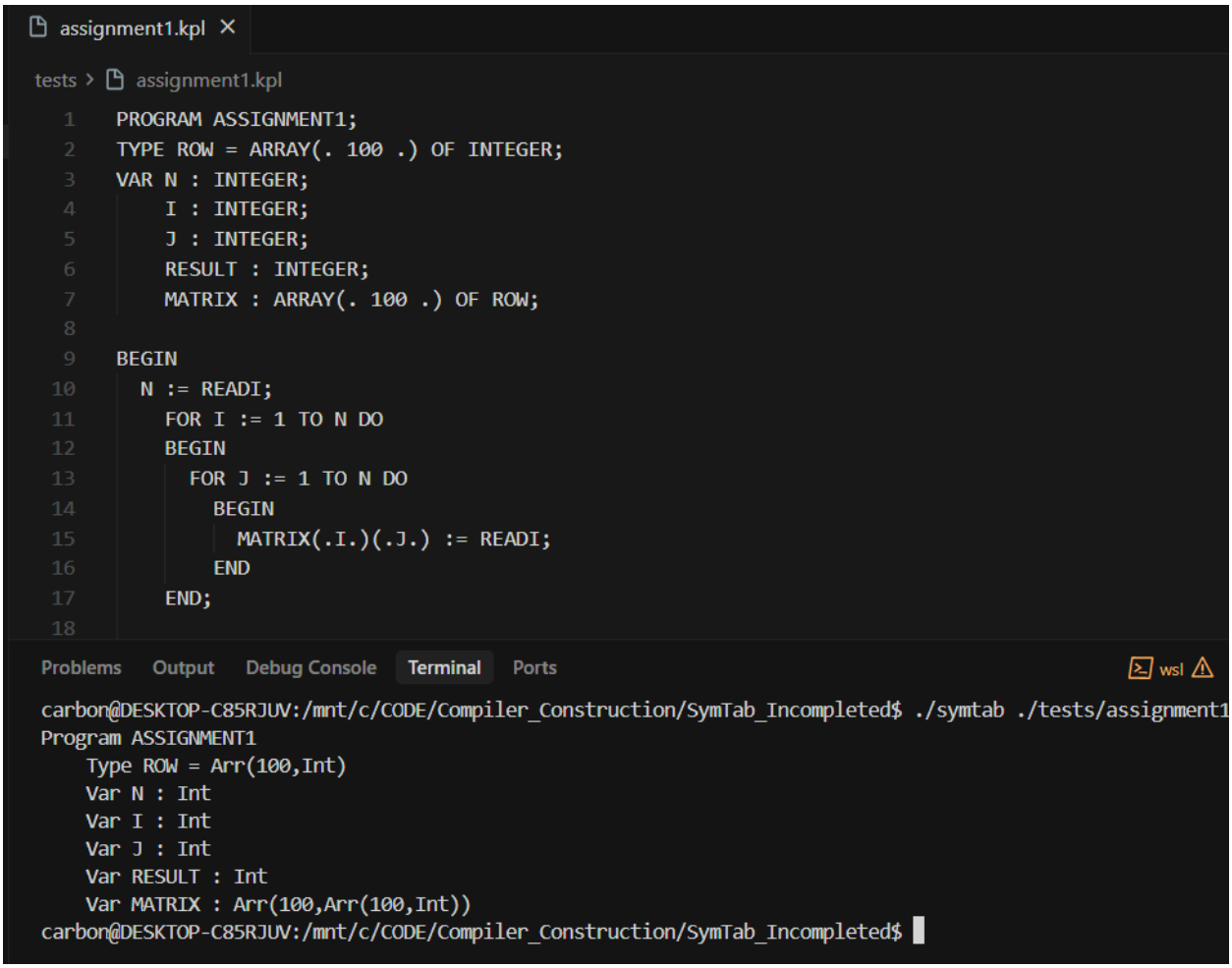
```
PROGRAM ASSIGNMENT1;
TYPE ROW = ARRAY(. 100 .) OF INTEGER;
VAR N : INTEGER;
    I : INTEGER;
    J : INTEGER;
    RESULT : INTEGER;
    MATRIX : ARRAY(. 100 .) OF ROW;

BEGIN
  N := READI;
    FOR I := 1 TO N DO
    BEGIN
      FOR J := 1 TO N DO
        BEGIN
          MATRIX(.I.)(.J.) := READI;
        END
    END;

  RESULT := 1;
  I := 2;
  WHILE I <= N DO
    BEGIN
      J := 1;
      WHILE J < I DO
        BEGIN
          IF MATRIX(.I.)(.J.) != 0 THEN
            BEGIN
              RESULT := 0;
            END;
          J := J + 1;
        END;
      I := I + 1;
    END;

  CALL WRITEI(RESULT);
  CALL WRITELN;
END.
```

Kết quả:



Giải thích:

Khi parser xử lý file assignment1.kpl, symbol table được xây dựng như sau:

1. Program ASSIGNMENT1: Tên chương trình được tạo trong symbol table.
2. Các khai báo toàn cục (indent 4 spaces) — được khai báo ở dòng 2-7:

- Type ROW = Arr(100,Int): Kiểu ROW là mảng 100 phần tử kiểu INTEGER (dòng 2: TYPE ROW = ARRAY(. 100 .) OF INTEGER;)
- Var N : Int: Biến N kiểu INTEGER (dòng 3)
- Var I : Int: Biến I kiểu INTEGER (dòng 4)
- Var J : Int: Biến J kiểu INTEGER (dòng 5)
- Var RESULT : Int: Biến RESULT kiểu INTEGER (dòng 6)

- Var MATRIX : Arr(100,Arr(100,Int)): Biến MATRIX là mảng 100 phần tử, mỗi phần tử là mảng 100 phần tử kiểu INTEGER (dòng 7: VAR MATRIX : ARRAY(. 100 .) OF ROW; với ROW = Arr(100,Int), nên MATRIX là mảng 2 chiều 100x100)

Tất cả thuộc scope chương trình, nên có indent 4 spaces.

2. Bài tập 2

Bài 2

Viết chương trình tính và in ra tổng 2 số bằng ngôn ngữ KPL. Chỉ ra phân tích trái của chương trình (dãy số hiệu sản xuất được dùng trong suy dẫn trái)

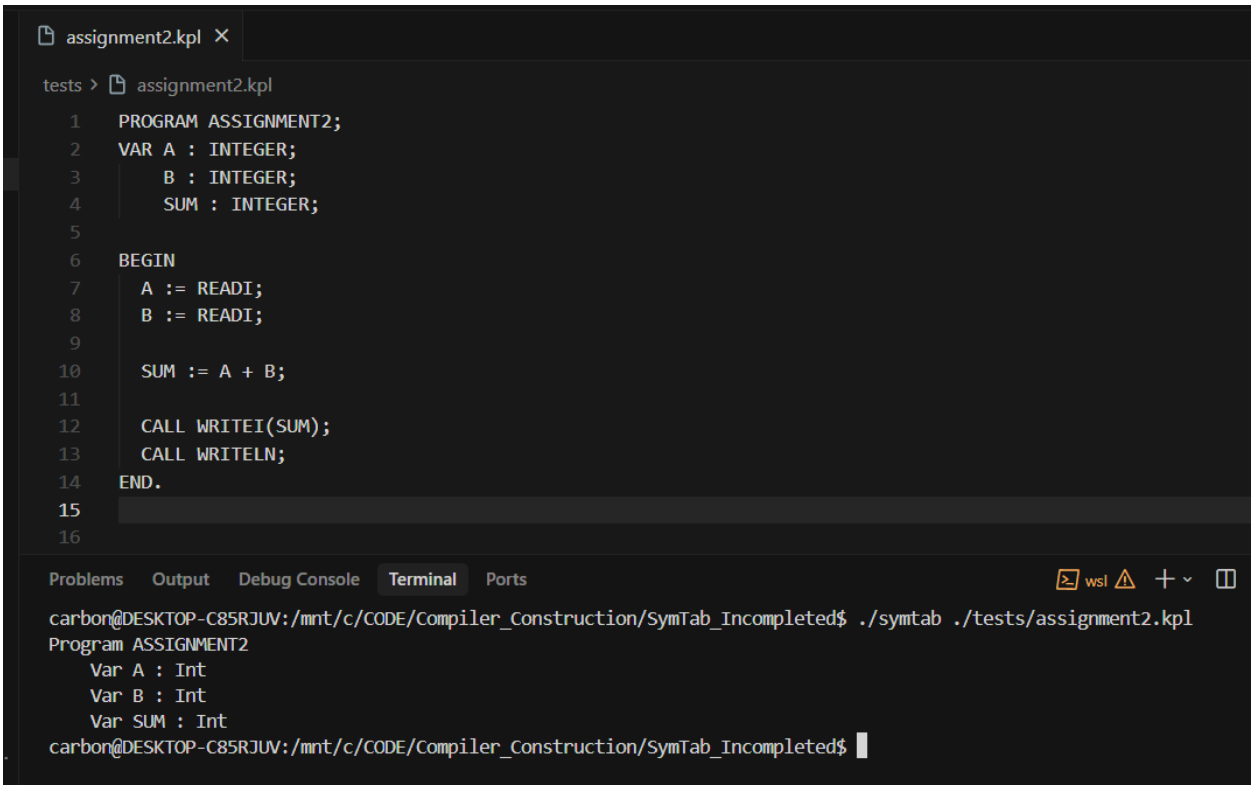
Chương trình:

assignment2.kpl

tests > assignment2.kpl

```
1   PROGRAM ASSIGNMENT2;
2   VAR A : INTEGER;
3       B : INTEGER;
4       SUM : INTEGER;
5
6   BEGIN
7     A := READI;
8     B := READI;
9
10    SUM := A + B;
11
12    CALL WRITEI(SUM);
13    CALL WRITELN;
14  END.
```

Kết quả:



Giải thích:

Khi parser xử lý file assignment2.kpl, symbol table được xây dựng như sau:

1. Program ASSIGNMENT2: Tên chương trình được tạo trong symbol table.
2. Các biến toàn cục (indent 4 spaces) — được khai báo ở dòng 2-4:

- Var A : Int: Biến A kiểu INTEGER (dòng 2: VAR A : INTEGER;)
- Var B : Int: Biến B kiểu INTEGER (dòng 3: VAR B : INTEGER;)
- Var SUM : Int: Biến SUM kiểu INTEGER (dòng 4: VAR SUM : INTEGER;)

Tất cả thuộc scope chương trình, nên có indent 4 spaces.