

BÁO CÁO THỰC HÀNH XÂY DỰNG CHƯƠNG TRÌNH DỊCH

Bài 3: Tạo bảng ký hiệu

Họ và tên: Trịnh Hữu An

MSSV: 20225593

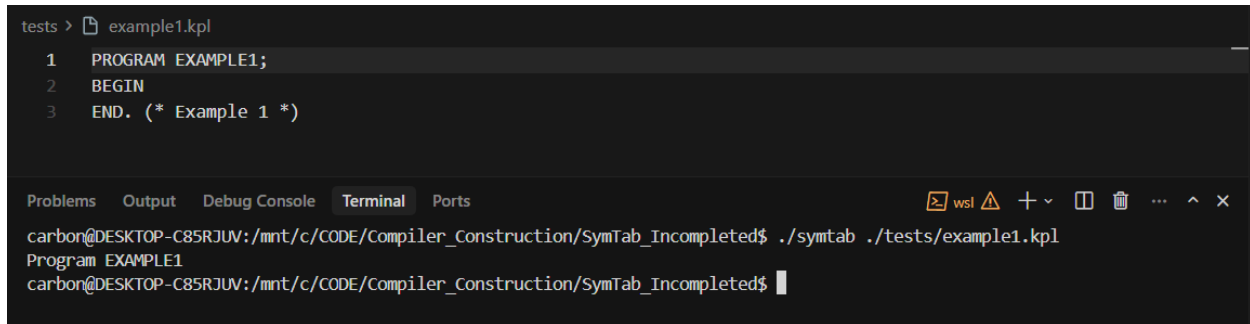
Mã lớp: 161629

Mục lục:

I.	Kết quả thực hiện với các ví dụ:	2
1.	Kết quả với example1.kpl	2
2.	Kết quả với example2.kpl	3
3.	Kết quả với example3.kpl	4
4.	Kết quả với example4.kpl	6
5.	Kết quả với example5.kpl	8
6.	Kết quả với example6.kpl	9
II.	Thực hiện với chương trình KPL ở phần lý thuyết:.....	11
1.	Bài tập 1.....	11
2.	Bài tập 2.....	14

I. Kết quả thực hiện với các ví dụ:

1. Kết quả với example1.kpl



The screenshot shows a code editor with the following content:

```
1 PROGRAM EXAMPLE1;  
2 BEGIN  
3 END. (* Example 1 *)
```

Below the editor is a terminal window with the following output:

```
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$ ./symtab ./tests/example1.kpl  
Program EXAMPLE1  
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$
```

Giải thích:

- 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 .
- 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)
- 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

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:

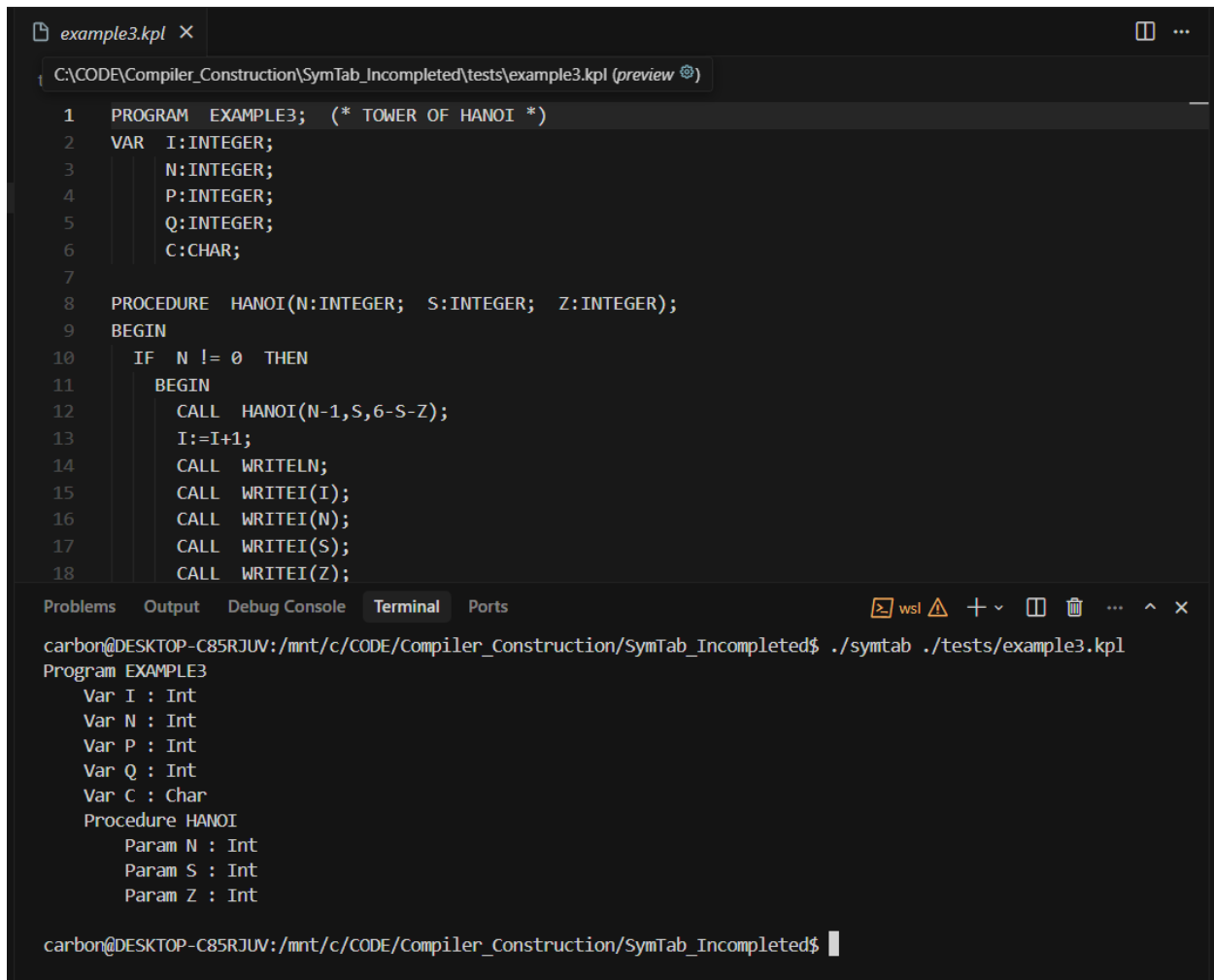
1. Program EXAMPLE2: Tên chương trình được tạo trong symbol table.
2. Var N : Int (indent 4 spaces): Biến N kiểu INTEGER được khai báo ở dòng 3 (VAR N : INTEGER;), thuộc scope của chương trình. Indent 4 spaces cho biết đây là object trong scope chương trình.
3. Function F : Int (indent 4 spaces): Hàm 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



```
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;
14 CALL WRITELN;
15 CALL WRITEI(I);
16 CALL WRITEI(N);
17 CALL WRITEI(S);
18 CALL WRITEI(Z);
```

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

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

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

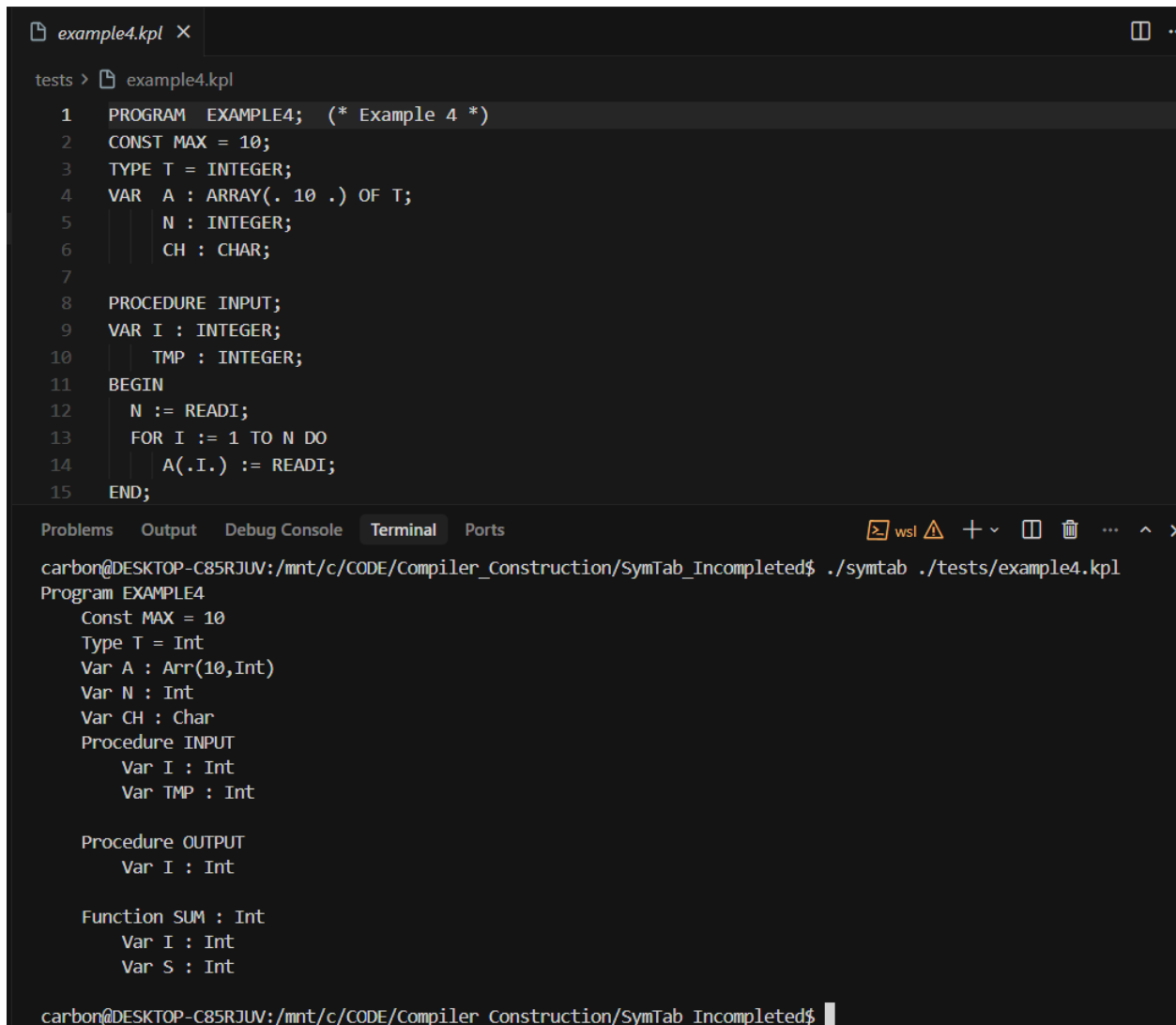
Tất cả thuộc scope của procedure HANOI, nên có indent 8 spaces.

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



```
example4.kpl x
tests > example4.kpl
1 PROGRAM EXAMPLE4; (* Example 4 *)
2 CONST MAX = 10;
3 TYPE T = INTEGER;
4 VAR A : ARRAY(. 10 .) OF T;
5     N : INTEGER;
6     CH : CHAR;
7
8 PROCEDURE INPUT;
9 VAR I : INTEGER;
10    TMP : INTEGER;
11 BEGIN
12     N := READI;
13     FOR I := 1 TO N DO
14         A(.I.) := READI;
15 END;
```

```
Problems Output Debug Console Terminal Ports
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$ ./symtab ./tests/example4.kpl
Program EXAMPLE4
  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

  Procedure OUTPUT
    Var I : Int

  Function SUM : Int
    Var I : Int
    Var S : Int

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

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



```
example5.kpl x
tests > example5.kpl
1 PROGRAM EXAMPLE5; (* Example 5 *)
2 CONST C = 1;
3 TYPE T = CHAR;
4 FUNCTION F(I : INTEGER):CHAR;
5 CONST B = C;
6 TYPE A = ARRAY(.5.) OF T;
7 BEGIN
8 END;
9 BEGIN
10 END. (* Example 5 *)

Problems Output Debug Console Terminal Ports
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$ ./symtab ./tests/example5.kpl
Program EXAMPLE5
  Const C = 1
  Type T = Char
  Function F : Char
    Param I : Int
    Const B = 1
    Type A = Arr(5,Char)

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

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

```
tests > example6.kpl

1 PROGRAM EXAMPLE6;
2   CONST C1 = 10;
3     C2 = 'a';
4   TYPE T1 = ARRAY(. 10 .) OF INTEGER;
5   VAR V1 : INTEGER;
6     V2 : ARRAY(. 10 .) OF T1;
7
8   FUNCTION F(P1 : INTEGER; VAR P2 : CHAR) : INTEGER;
9     BEGIN
10      F := C1;
```

Problems Output Debug Console **Terminal** Ports

```
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$ ./symtab ./tests/example6.kpl
Program EXAMPLE6
  Const C1 = 10
  Const C2 = 'a'
  Type T1 = Arr(10,Int)
  Var V1 : Int
  Var V2 : Arr(10,Arr(10,Int))
  Function F : Int
    Param P1 : Int
  Function F : Int
  Function F : Int
    Param P1 : Int
    Param VAR P2 : Char

  Procedure P
    Param V1 : Int
    Const C1 = 'a'
    Const C3 = 10
    Type T1 = Int
    Type T2 = Arr(10,Int)
    Var V2 : Arr(10,Int)
    Var V3 : Char

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

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 $n \times n$, 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:

Upper triangular matrix: U

1	1/2	3	0
0	5	0	1
0	0	4	-2
0	0	0	3

Chương trình:

```
assignment1.kpl X
tests > assignment1.kpl
1  PROGRAM ASSIGNMENT1;
2  TYPE ROW = ARRAY(. 100 .) OF INTEGER;
3  VAR N : INTEGER;
4      I : INTEGER;
5      J : INTEGER;
6      RESULT : INTEGER;
7      MATRIX : ARRAY(. 100 .) OF ROW;
8
9  BEGIN
10     N := READI;
11     FOR I := 1 TO N DO
12         BEGIN
13             FOR J := 1 TO N DO
14                 BEGIN
15                     MATRIX(.I.)(.J.) := READI;
16                 END
17             END;
18
19     RESULT := 1;
20     I := 2;
21     WHILE I <= N DO
22         BEGIN
23             J := 1;
24             WHILE J < I DO
25                 BEGIN
26                     IF MATRIX(.I.)(.J.) != 0 THEN
27                         BEGIN
28                             RESULT := 0;
29                         END;
30                     J := J + 1;
31                 END;
32             I := I + 1;
33         END;
34
35     CALL WRITEI(RESULT);
36     CALL Writeln;
37 END.
```

Kết quả:

```
assignment1.kpl X
tests > assignment1.kpl
1  PROGRAM ASSIGNMENT1;
2  TYPE ROW = ARRAY(. 100 .) OF INTEGER;
3  VAR N : INTEGER;
4      I : INTEGER;
5      J : INTEGER;
6      RESULT : INTEGER;
7      MATRIX : ARRAY(. 100 .) OF ROW;
8
9  BEGIN
10     N := READI;
11     FOR I := 1 TO N DO
12     BEGIN
13         FOR J := 1 TO N DO
14         BEGIN
15             MATRIX(.I.)(.J.) := READI;
16         END
17     END;
18
Problems  Output  Debug Console  Terminal  Ports
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$ ./symtab ./tests/assignment1
Program ASSIGNMENT1
  Type ROW = Arr(100,Int)
  Var N : Int
  Var I : Int
  Var J : Int
  Var RESULT : Int
  Var MATRIX : Arr(100,Arr(100,Int))
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$
```

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 X
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:

The screenshot shows a code editor with a file named `assignment2.kpl` open. The code is as follows:

```
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.
15
16
```

Below the code editor is a terminal window. The terminal shows the command `./symtab ./tests/assignment2.kpl` being executed. The output of the command is:

```
Program ASSIGNMENT2
  Var A : Int
  Var B : Int
  Var SUM : Int
```

The terminal prompt is `carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/SymTab_Incompleted$`.

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.