BÁO CÁO THỰC HÀNH
XÂY DỰNG CHƯƠNG TRÌNH DỊCH
BÀI 3: TẠO BẢNG KÝ HIỆU

Họ và tên: Bùi Quang Hưng

Mã số sinh viên: 20225849

I. Tổng quan dự án

1. Mục đích

- Dự án là một trình phân tích cú pháp (parser) và quản lý bảng ký hiệu (symbol table) cho ngôn ngữ lập trình đơn giản (KPL). Mục tiêu là nhận diện, phân tích cú pháp, kiểm tra ngữ nghĩa và xây dựng bảng ký hiệu cho chương trình nguồn, phục vụ cho các bước biên dịch tiếp theo.

2. Luồng hoạt động

a) Khởi động chương trình

- Hàm main nhận tên file nguồn (ví dụ: example1.kpl) từ dòng lệnh.

- Nếu không có file, in ra thông báo lỗi và kết thúc.

- Gọi hàm compile với tên file nguồn.

b) Tiền xử lý file nguồn

- Hàm **openInputStream** mở file nguồn, khởi tạo biến toàn cục lineNo, colNo và đọc ký tự đầu tiên vào currentChar.

- Các hàm readChar sẽ đọc từng ký tự tiếp theo, cập nhật số dòng/cột.

c) Phân tích từ vựng (Lexical Analysis)

- Sử dụng mảng charCodes để phân loại ký tự (chữ, số, toán tử, v.v.).

- Hàm **getToken** thực hiện máy trạng thái để nhận diện token (từ khóa, số, tên biến, toán tử, ký tự đặc biệt...).

- Hàm **getValidToken** bỏ qua các token không hợp lệ.

- Token được biểu diễn bởi struct Token.

d) Phân tích cú pháp (Syntax Analysis)

- Hàm **compile** là điểm vào chính:

- Gọi **openInputStream** để mở file.

- Lấy token đầu tiên qua **getValidToken**.

- Khởi tạo bảng ký hiệu (symbol table) qua **initSymTab**.

- Gọi **compileProgram** để bắt đầu phân tích cú pháp chương trình.

- Sau khi phân tích xong, in ra cấu trúc bảng ký hiệu qua **printObject**.

- Giải phóng bộ nhớ và đóng file.

- Cấu trúc phân tích cú pháp

- Các hàm chính:

  + **compileProgram**: Phân tích chương trình chính (PROGRAM ...).
  + **compileBlock**, **compileBlock2**, ...: Phân tích các khối khai báo hằng, kiểu, biến, hàm, thủ tục.
  + **compileStatements**, **compileStatement**: Phân tích các câu lệnh.

- Các hàm con khác phân tích biểu thức, điều kiện, tham số, v.v.

- Cách hoạt động:

  + Sử dụng hàm eat để kiểm tra và lấy token tiếp theo, đảm bảo đúng cú pháp.
  + Nếu gặp lỗi cú pháp, gọi **missingToken**(error.c) hoặc error để in thông báo và kết thúc chương trình.

e) Bảng ký hiệu (Symbol Table)

- Quản lý các đối tượng: chương trình, hằng, biến, kiểu, hàm, thủ tục, tham số.

- Khi gặp khai báo mới, tạo đối tượng tương ứng (ví dụ: createConstantObject), lưu vào bảng ký hiệu hiện tại.

- Hỗ trợ phạm vi lồng nhau (scope) cho hàm/thủ tục.

- Khi vào khối mới, gọi **enterBlock** (symtab.c); khi ra khỏi khối, gọi **exitBlock**.

- Khi cần tra cứu tên, dùng **lookupObject**.

f) In cấu trúc bảng ký hiệu

- Sau khi phân tích xong, chương trình sẽ in ra cấu trúc bảng ký hiệu (các hằng, biến, kiểu, hàm, thủ tục, tham số...) theo dạng cây, giúp kiểm tra kết quả phân tích.

g) Xử lý lỗi

- Khi phát hiện lỗi (cú pháp, khai báo, kiểu...), chương trình sẽ in ra thông báo lỗi kèm vị trí (dòng, cột) và kết thúc.

II. Các case đã implement

a) Khai báo chương trình: Từ khóa **PROGRAM**, tên chương trình, dấu chấm phẩy (;), khối chương trình, dấu chấm kết thúc.

b) Khai báo hằng số: Từ khóa **CONST**, khai báo nhiều hằng số với giá trị số/nguyên/biến đã khai báo trước.

c) Khai báo kiểu: Từ khóa **TYPE**, khai báo kiểu cơ bản (INTEGER, CHAR), kiểu mảng, kiểu tự định nghĩa.

d) Khai báo biến: Từ khóa **VAR**, khai báo nhiều biến với kiểu xác định.

e) Khai báo hàm/thủ tục: Từ khóa **FUNCTION**/**PROCEDURE**, khai báo tên, tham số, kiểu trả về (hàm), khối lệnh.

f) Khai báo tham số: Tham số truyền giá trị hoặc tham chiếu (VAR), kiểu cơ bản.

g) Biểu thức, điều kiện, lệnh gán, lệnh gọi hàm/thủ tục, lệnh nhóm, lệnh điều kiện (IF), vòng lặp (WHILE, FOR).

h) Quản lý scope: Tạo scope cho chương trình, hàm, thủ tục; quản lý vào/ra scope.

i) Tra cứu và kiểm tra đối tượng: Kiểm tra khai báo trùng, tra cứu đối tượng theo scope, kiểm tra kiểu, giá trị hằng số.

k) In bảng ký hiệu: In ra cấu trúc bảng ký hiệu sau khi phân tích xong.

III. Kết quả

1. Thực hiện chương trình với các tệp example có sẵn

- Example 1:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe tests\example1.kpl
Program EXAMPLE1
PS D:\CTD_Lab\SymTab_Incompleted>
```

- Example 2:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe tests\example2.kpl
Program EXAMPLE2
    Var N : Int
    Function F : Int
        Param N : Int

PS D:\CTD_Lab\SymTab_Incompleted>
```

- Example 3:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe 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

PS D:\CTD_Lab\SymTab_Incompleted>
```

- Example 4:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe 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
```

- Example 5:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe 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)

PS D:\CTD_Lab\SymTab_Incompleted>
```

- Example 6:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe 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
        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

PS D:\CTD_Lab\SymTab_Incompleted>
```

2. Thực hiện chương trình với bài tập lý thuyết

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:

Upper triangular matrix: U

$$\begin{bmatrix} 1 & 1/2 & 3 & 0 \\ 0 & 5 & 0 & 1 \\ 0 & 0 & 4 & -2 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

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)

- Bài 1:

+ Mã nguồn chương trình kpl:

tests > matrix_upper_triangular.kpl

```
PROGRAM MATRIX;

CONST MAXN = 10;

VAR N : INTEGER;
    I : INTEGER;
    J : INTEGER;
    A : ARRAY(. 10 .) OF ARRAY(. 10 .) OF INTEGER;
    ISUPPER : INTEGER;

PROCEDURE INPUT;
VAR I : INTEGER;
    J : INTEGER;
BEGIN
    FOR I := 1 TO N DO
        FOR J := 1 TO N DO
            A(.I.)(.J.) := READI
END;

FUNCTION CHECKUPPER : INTEGER;
VAR I : INTEGER;
    J : INTEGER;
    RESULT : INTEGER;
BEGIN
    RESULT := 1;
    FOR I := 1 TO N DO
        FOR J := 1 TO N DO
            IF I > J THEN
                IF A(.I.)(.J.) != 0 THEN
                    RESULT := 0;
    CHECKUPPER := RESULT
END;

BEGIN
    CALL WRITELN;
    (* Nhap kich thuoc ma tran *)
    N := READI;

    (* Nhap ma tran *)
    CALL INPUT;

    (* Kiem tra va in ket qua *)
    ISUPPER := CHECKUPPER;
    CALL WRITELN;
    CALL WRITEI(ISUPPER);
    CALL WRITELN
END.
```

+ Kết quả:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe tests\matrix_upper_triangular.kpl
Program MATRIX
    Const MAXN = 10
    Var N : Int
    Var I : Int
    Var J : Int
    Var A : Arr(10,Arr(10,Int))
    Var ISUPPER : Int
    Procedure INPUT
        Var I : Int
        Var J : Int

    Function CHECKUPPER : Int
        Var I : Int
        Var J : Int
        Var RESULT : Int

PS D:\CTD_Lab\SymTab_Incompleted>
```

- Bài 2:

+ Mã nguồn chương trình kpl:

tests > sum_two_numbers.kpl

```
PROGRAM SUMTWONUMBERS;

VAR A : INTEGER;
    B : INTEGER;
    SUM : INTEGER;

FUNCTION ADD(X : INTEGER; Y : INTEGER) : INTEGER;
BEGIN
    ADD := X + Y
END;

BEGIN
    CALL WRITELN;

    (* Nhap so thu nhat *)
    A := READI;

    (* Nhap so thu hai *)
    B := READI;

    (* Tinh tong *)
    SUM := ADD(A, B);

    (* In ket qua *)
    CALL WRITELN;
    CALL WRITEI(SUM);
    CALL WRITELN
END.

```

+ Kết quả:

```
PS D:\CTD_Lab\SymTab_Incompleted> .\parser.exe tests\sum_two_numbers.kpl
Program SUMTWONUMBERS
    Var A : Int
    Var B : Int
    Var SUM : Int
    Function ADD : Int
        Param X : Int
        Param Y : Int

PS D:\CTD_Lab\SymTab_Incompleted>
```