BÁO CÁO THỰC HÀNH
XÂY DỰNG CHƯƠNG TRÌNH DỊCH
BÀI 1: PHÂN TÍCH TỪ VỰNG

Họ và tên: Bùi Quang Hưng

Mã số sinh viên: 20225849

I. TỔNG QUAN DỰ ÁN

1. Mục đích

Dự án là một scanner cho ngôn ngữ KPL (ngôn ngữ lập trình giáo dục giống Pascal). Scanner sẽ đọc file mã nguồn .kpl, phân tích từng ký tự, nhóm thành các token (từ khóa, tên biến, số, toán tử, dấu câu,...) và in ra thông tin token theo định dạng dòng-cột:loại_token.

2. Luồng hoạt động

a) Khởi động

- Chạy chương trình với tham số là tên file .kpl: scanner.exe example1.kpl
- Hàm main() kiểm tra tham số đầu vào và gọi hàm scan()

b) Quét file

- Hàm scan() mở file bằng openInputStream(), khởi tạo state = 0
- Gọi getToken() liên tục cho đến khi gặp token TK_EOF
- Mỗi token được in ra màn hình bằng printToken()

c) Phân tích

Scanner sử dụng máy trạng thái hữu hạn **(Finite State Machine)** với biến state để điều khiển luồng xử lý:

- **Bỏ qua khoảng trắng**: spaces, tabs, newlines
- **Nhận diện từ khóa/tên biến**: gom các ký tự chữ/số,
  dùng checkKeyword() phân biệt keyword và identifier
- **Nhận diện số**: gom các ký tự số liên tiếp
- **Nhận diện toán tử đơn**: +, -, *, /, =, ,, ., ;, :, (,)
- **Nhận diện toán tử ghép**: <=, >=, :=, !=

- **Nhận diện hằng ký tự**: 'x' (1 ký tự trong dấu nháy đơn)
- **Xử lý comment**: (* ... *)
- **Xử lý lỗi**: ký tự không hợp lệ, tên biến quá dài, hằng ký tự sai, comment chưa đóng

d) In token

- In ra dòng-cột và loại token, nếu là tên biến/số/hằng thì thêm giá trị

II. CÁC CASE ĐÃ IMPLEMENT

1. Khởi động và điều hướng trạng thái

- Case 0: Trạng thái bắt đầu, kiểm tra loại ký tự hiện tại để chuyển sang state phù hợp:
    - Chữ cái → state 3 (identifier/keyword)
    - Chữ số → state 7 (number)
    - Dấu + → state 9, - → state 10, * → state 11, / → state 12
    - Dấu < → state 13, > → state 16, = → state 19, ! → state 20
    - Dấu , → state 23, . → state 24, ; → state 27, : → state 28
    - Dấu ' → state 31 (char literal)
    - Dấu (→ state 35 (comment hoặc LPAR)
    - Dấu) → state 42
    - Ký tự không hợp lệ → state 43
    - EOF → state 1

2. Kết thúc file

- Case 1: Gặp EOF, tạo token TK_EOF và kết thúc

3. Bỏ qua khoảng trắng

- Case 2: Đọc ký tự tiếp theo, reset state về 0, tiếp tục quét

4. Nhận diện từ khóa và tên biến

- Case 3: Gom các ký tự chữ/số thành chuỗi:
    - Kiểm tra độ dài ≤ 15 ký tự
    - Nếu vượt quá → báo lỗi ERR_IDENTTOOLONG

    - Chuyển sang case 4
- Case 4: Dùng checkKeyword() phân biệt:
    - Nếu là keyword → tạo token keyword (KW_PROGRAM, KW_BEGIN,...)
    - Nếu không → tạo token TK_IDENT

5. Nhận diện số nguyên

- Case 7: Gom các ký tự số thành chuỗi, tạo token TK_NUMBER

6. Nhận diện các toán tử và ký hiệu đơn

- Case 9: Dấu + → SB_PLUS
- Case 10: Dấu - → SB_MINUS
- Case 11: Dấu * → SB_TIMES
- Case 12: Dấu / → SB_SLASH
- Case 19: Dấu = → SB_EQ
- Case 23: Dấu , → SB_COMMA
- Case 24-26: Dấu . → SB_PERIOD (hoặc . kết hợp) → SB_RSEL)
- Case 27: Dấu ; → SB_SEMICOLON
- Case 42: Dấu) → SB_RPAR

7. Nhận diện toán tử so sánh, gán và đặc biệt

- Case 13-15:
    - Dấu < → nếu theo sau là = thì SB_LE, ngược lại SB_LT
- Case 16-18:
    - Dấu > → nếu theo sau là = thì SB_GE, ngược lại SB_GT
- Case 20-22:
    - Dấu ! → nếu theo sau là = thì SB_NEQ, ngược lại báo lỗi ERR_INVALIDSYMBOL
- Case 28-30:
    - Dấu : → nếu theo sau là = thì SB_ASSIGN, ngược lại SB_COLON

8. Nhận diện hằng ký tự

- Case 31-34: Nhận diện ký tự dạng 'x':
    - Đọc ký tự đầu '
    - Đọc 1 ký tự in được
    - Đọc ký tự đóng '
    - Nếu không đúng format → báo lỗi ERR_INVALIDCHARCONSTANT

9. Nhận diện comment và ký hiệu đặc biệt

- Case 35-41: Xử lý dấu (:
    - Nếu sau (là * → bắt đầu comment
        - Đọc đến khi gặp *)
        - Nếu không gặp (EOF) → báo lỗi ERR_ENDOFCOMMENT
    - Nếu sau (là . → SB_LSEL (array selector)
    - Ngược lại → SB_LPAR

10. Ký hiệu không hợp lệ

- Case 43: Gặp ký tự không nhận diện được → báo lỗi ERR_INVALIDSYMBOL, tạo token TK_NONE

III. KẾT QUẢ

1. Thực hiện kết quả với file example1.kpl (không có lỗi)

```
D:\CTD_Lab\20225849_BuiQuangHung\Scanner>.\scanner.exe test\example1.kpl
1-1:KW_PROGRAM
1-9:TK_IDENT(Example1)
1-17:SB_SEMICOLON
2-1:KW_BEGIN
3-1:KW_END
3-4:SB_PERIOD
```

- **Nhận xét**: Scanner nhận diện đúng tất cả token, bỏ qua comment, không có lỗi.

2. Trường hợp có lỗi **ERR_INVALIDSYMBOL**

- Nội dung đầu vào:

```
Program ErrorTest;
Var
  (* This is a comment that never ends...
  x : Integer;
  verylongidentifiernamethatexceedsthemaximumlength : Integer;
  c : Char;
Begin
  c := 'ab';  (* Invalid char constant - too long *)
  x := 123;
  @ := 5;  (* Invalid symbol *)
End.
```

- Kết quả:

```
D:\CTD_Lab\20225849_BuiQuangHung\Scanner>.\scanner.exe test\example_errors.kpl
1-1:KW_PROGRAM
1-9:TK_IDENT(ErrorTest)
1-18:SB_SEMICOLON
2-1:KW_VAR
9-3:TK_IDENT(x)
9-5:SB_ASSIGN
9-8:TK_NUMBER(123)
9-11:SB_SEMICOLON
10-3:Invalid symbol!
10-3:TK_NONE
10-5:SB_ASSIGN
10-8:TK_NUMBER(5)
10-9:SB_SEMICOLON
11-1:KW_END
11-4:SB_PERIOD
```

- ERR_ENDOFCOMMENT: Comment bắt đầu từ dòng 3 không được đóng, scanner bỏ qua toàn bộ nội dung cho đến cuối file
- ERR_INVALIDCHARCONSTANT: Không xuất hiện vì code bị comment
- ERR_INVALIDSYMBOL: Phát hiện ký tự @ không hợp lệ tại dòng **10**, cột **3** và báo lỗi **Invalid symbol!**

- **Nhận xét**: Do lỗi comment chưa đóng ở đầu file, scanner đã bỏ qua phần lớn code, chỉ phát hiện được lỗi **invalid symbol** ở cuối. Điều này cho thấy scanner xử lý đúng theo thứ tự từ trên xuống.