



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

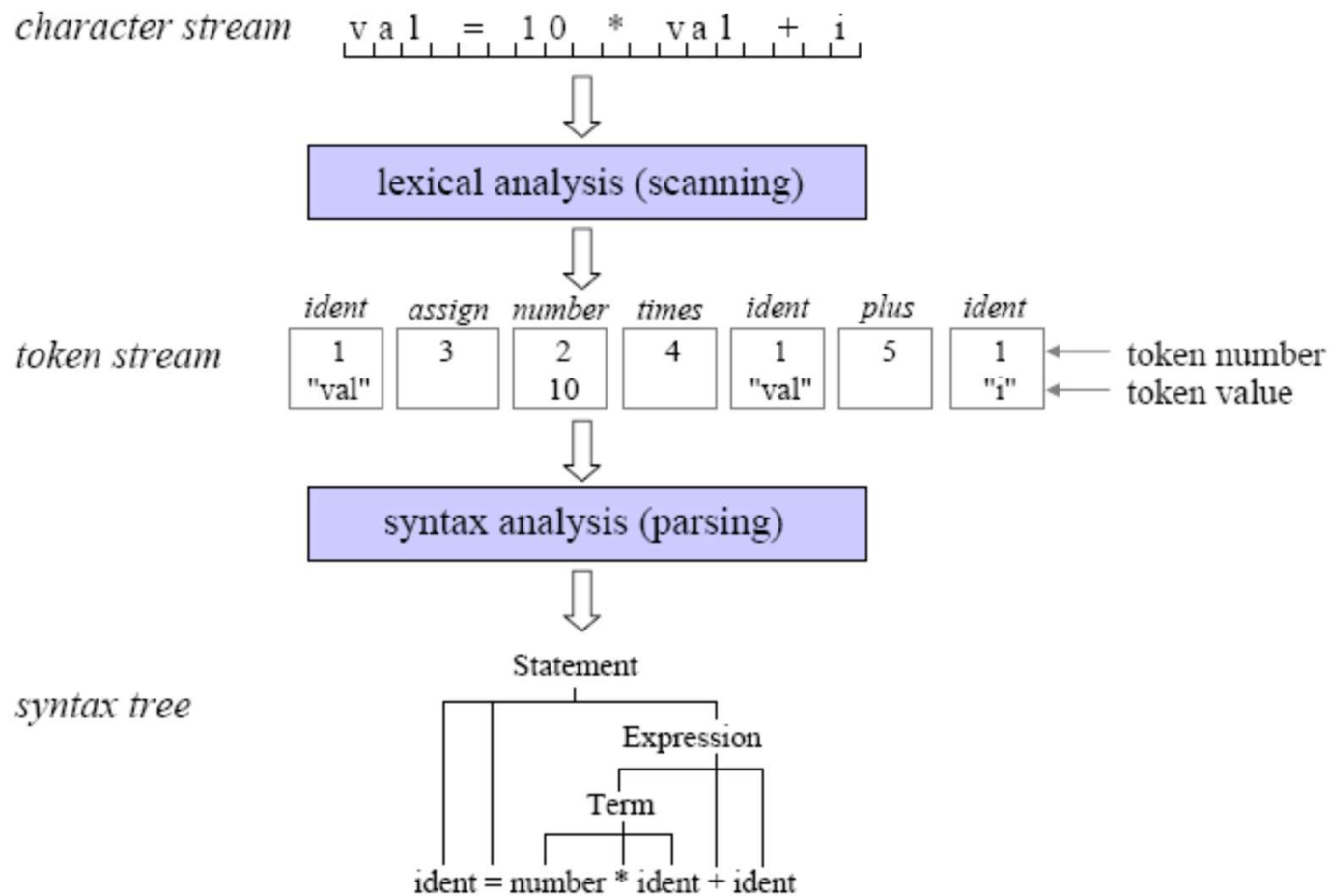
Thực hành Xây dựng chương trình dịch

Bài 1: Phân tích từ vựng

Scanner là gì?

- Trong một chương trình dịch, thành phần thực hiện chức năng phân tích từ vựng gọi là scanner.

Bộ phân tích từ vựng (scanner) là gì?



Nhiệm vụ của một scanner

- Bỏ qua các ký tự vô nghĩa như: dấu trống, tab, ký tự xuống dòng, chú thích.
- Phát hiện các ký tự không hợp lệ
- Phát hiện token
 - định danh (identifier)
 - từ khóa (keyword)
 - số (number)
 - Hằng ký tự
 - special symbol
 - ...

Nhiệm vụ của một scanner

- Chuyển lần lượt các token cho bộ phân tích cú pháp (parser)

Bảng chữ cái của KPL

- Chữ cái (letter): a-z, A-Z, ‘_’
- Chữ số (digit): 0-9
- Các ký hiệu đặc biệt
 - +, -, *, /, >, <, !, =, [space], [comma], ., :, ;, ‘, (,)

Các token của ngôn ngữ KPL

- Từ khóa

PROGRAM, CONST, TYPE, VAR, PROCEDURE, FUNCTION, BEGIN, END, ARRAY, OF, INTEGER, CHAR, CALL, IF, ELSE, WHILE, DO, FOR, TO

- Toán tử

:= (assign), + (addition), - (subtraction), * (multiplication), / (division), = (comparison of equality), != (comparison of difference), > (comparison of greatness), < (comparison of lessness), >= (comparison of greatness or equality), <= (comparison of lessness or equality)

Các từ tổ của ngôn ngữ KPL

- Ký hiệu đặc biệt
; (semicolon), . (period), : (colon), , (comma), ((left parenthesis),) (right parenthesis), ‘ (singlequote)
- Và
(. và .) để đánh dấu chỉ mục của mảng
(* và *) để đánh dấu điểm bắt đầu và kết thúc của chú thích
- Ngoài ra
định danh, số, hằng ký tự

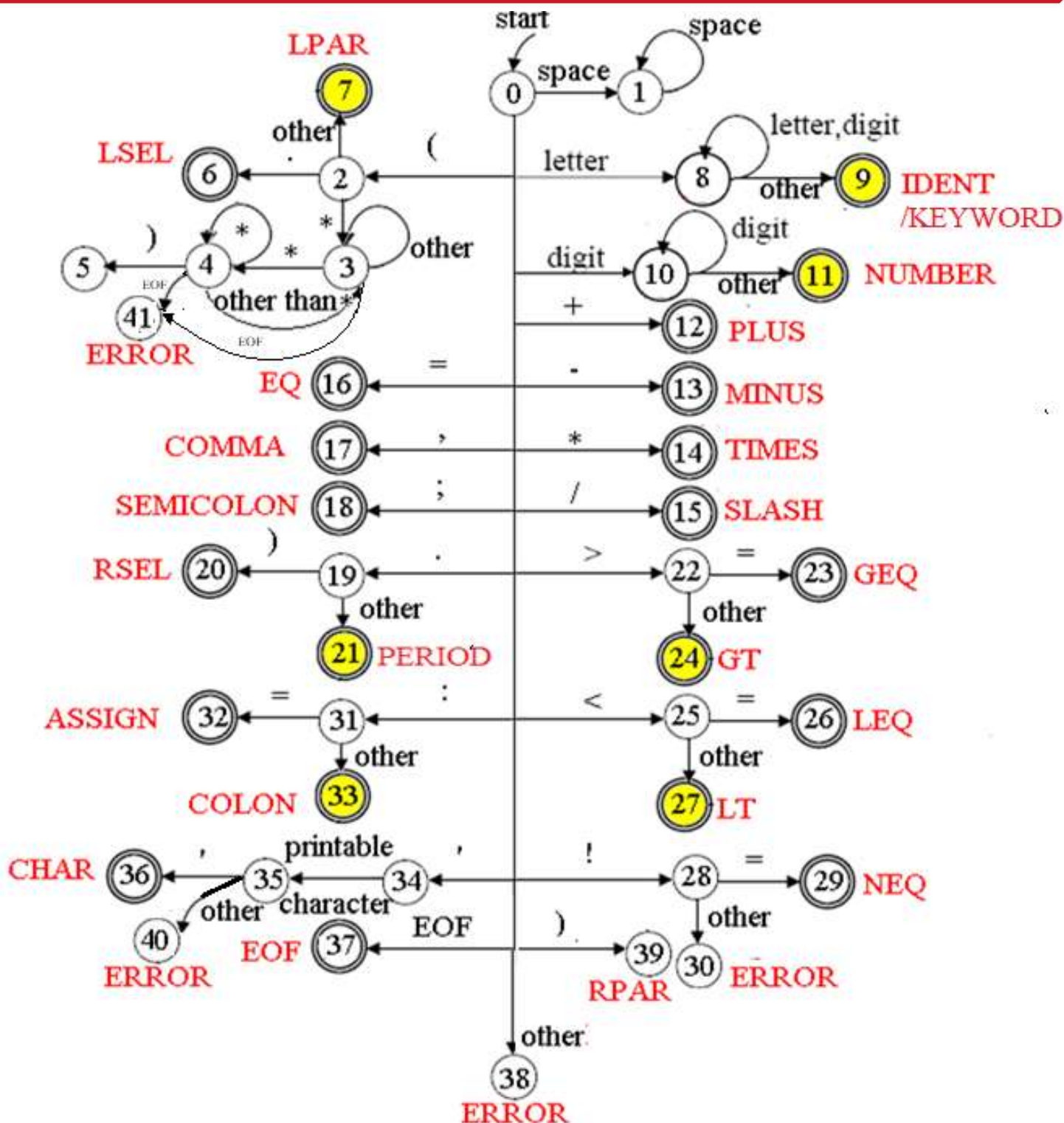
Nhận dạng các token của KPL

- Các token của KPL tạo nên một ngôn ngữ chính quy và có thể mô tả bởi một biểu thức chính quy.
- Chúng có thể nhận dạng bằng một automata hữu hạn đơn định
- scanner là một automata hữu hạn đơn định

Nhận dạng các token của KPL

Mỗi khi hoàn tất nhận dạng một token, automata sẽ chuyển lại về trạng thái 0

Khi có lỗi xảy ra (gặp ký tự ngoài bảng chữ cái,...), automata sẽ thông báo lỗi



Đầu vào, đầu ra của scanner

Vào

Program Example1; (*
Example 1*)

Begin

End. (* Example1*)

Ra

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

Xây dựng scanner – Cấu trúc

STT	Tên tệp	Nội dung
1	Makefile	Project
2	scanner.c	Tập chính
3	reader.h, reader.c	Đọc mã nguồn
4	charcode.h, charcode.c	Phân loại ký tự
5	token.h, token.c	Phân loại và nhận dạng token, từ khóa
6	error.h, error.c	Thông báo lỗi

Xây dựng scanner – reader

```
// Đọc một ký tự từ kênh vào
int readChar(void);
// Mở kênh vào
int openInputStream(char *fileName);
// Đóng kênh vào
void closeInputStream(void);

// Chỉ số dòng, cột hiện tại
int lineNo, colNo;
// Ký tự hiện tại
int currentChar;
```

Xây dựng scanner – charcode

```
typedef enum {  
    CHAR_SPACE,           // Khoảng trống  
    CHAR_LETTER,          // Chữ cái  
    CHAR_DIGIT,           // Chữ số  
    CHAR_PLUS,            // '+'  
    CHAR_MINUS,           // '-'  
    CHAR_TIMES,           // '*'  
    CHAR_SLASH,           // '/'  
    CHAR_LT,              // '<'  
    CHAR_GT,              // '>'  
    CHAR_EXCLAMATION,     // '!'  
    CHAR_EQ,              // '='  
    CHAR_COMMA,           // ','  
    CHAR_PERIOD,          // '.'  
    CHAR_COLON,           // ':'  
    CHAR_SEMICOLON,       // ';'  
    CHAR_SINGLEQUOTE,     // '\''  
    CHAR_LPAR,            // '('  
    CHAR_RPAR,            // ')'  
    CHAR_UNKNOWN          // Ký tự ngoài bảng chữ cái  
} CharCode;
```

Xây dựng scanner – charcode

- charcode.c định nghĩa một bảng charCodes ánh xạ từng ký tự trong bảng mã ASCII vào một trong các CharCode được định nghĩa
- Lưu ý: Lệnh đọc ký tự `getc` có thể trả về mã EOF có giá trị nguyên là -1, nằm ngoài bảng mã ASCII

Xây dựng scanner – token

```
typedef enum {  
    TK_NONE,           // Đại diện cho một lỗi  
    TK_IDENT,          // Định danh  
    TK_NUMBER,         // Số  
    TK_CHAR,           // Hằng ký tự  
    TK_EOF,            // Kết thúc chương trình  
    // Các từ khóa  
    KW_PROGRAM, KW_CONST, KW_TYPE, KW_VAR,  
    KW_INTEGER, KW_CHAR, KW_ARRAY, KW_OF,  
    KW_FUNCTION, KW_PROCEDURE,  
    KW_BEGIN, KW_END, KW_CALL,  
    KW_IF, KW_THEN, KW_ELSE,  
    KW_WHILE, KW_DO, KW_FOR, KW_TO,  
    // Các ký hiệu đặc biệt  
    SB_SEMICOLON, SB_COLON, SB_PERIOD, SB_COMMA,  
    SB_ASSIGN, SB_EQ, SB_NEQ, SB_LT, SB_LE, SB_GT, SB_GE,  
    SB_PLUS, SB_MINUS, SB_TIMES, SB_SLASH,  
    SB_LPAR, SB_RPAR, SB_LSEL, SB_RSEL  
} TokenType;
```


Xây dựng scanner – token

```
// Cấu trúc lưu trữ của một token
typedef struct {
    char string[MAX_IDENT_LEN + 1];
    int lineNo, colNo;
    TokenType tokenType;
    int value;
} Token;
```

```
// Kiểm tra một chuỗi có là từ khóa không
TokenType checkKeyword(char *string);
// Tạo một token mới với kiểu và vị trí
Token* makeToken(TokenType tokenType, int lineNo, int colNo);
```

Xây dựng scanner – error

```
// Danh sách các lỗi trong quá trình phân tích từ vựng
typedef enum {
    ERR_ENDOFCOMMENT,
    ERR_IDENTTOOLONG,
    ERR_INVALIDCHARCONSTANT,
    ERR_INVALIDSYMBOL
} ErrorCode;

// Các thông báo lỗi
#define ERM_ENDOFCOMMENT "End of comment expected!"
#define ERM_IDENTTOOLONG "Identification too long!"
#define ERM_INVALIDCHARCONSTANT "Invalid const char!"
#define ERM_INVALIDSYMBOL "Invalid symbol!"

// Hàm thông báo lỗi
void error(ErrorCode err, int lineNo, int colNo);
```

Xây dựng scanner – hàm xử lý chính

```
// Đọc một token tính từ vị trí hiện tại
Token* getToken(void) {
    Token *token;
    int ln, cn;

    if (currentChar == EOF)
        return makeToken(TK_EOF, lineNo, colNo);

    switch (charCodes[currentChar]) {
    case CHAR_SPACE: skipBlank(); return getToken();
    case CHAR_LETTER: return readIdentKeyword();
    case CHAR_DIGIT: return readNumber();
    case CHAR_PLUS:
        token = makeToken(SB_PLUS, lineNo, colNo);
        readChar();
        return token;
    ...
    }
```

Nhiệm vụ

- Hoàn thiện các hàm sau trong `scanner.c`
 - `void skipBlank();`
 - `void skipComment();`
 - `Token* readIdentKeyword(void);`
 - `Token* readNumber(void);`
 - `Token* readConstChar(void);`
 - `Token* getToken(void);`

Ví dụ

```
void skipBlank() {  
    while ((currentChar != EOF) &&  
        (charCodes[currentChar] == CHAR_SPACE))  
        readChar();  
}
```

Lưu ý:

- Quan tâm đến việc phân biệt chữ hoa/chữ thường
- Xử lý dấu _
- Độ dài số hợp lý để tránh lỗi khi chuyển từ ký tự sang số
- Không dừng chương trình khi gặp lỗi
- Nếu dùng ‘\’, ‘\\’ để biểu diễn các hằng ‘ và \ thì xử lý như thế nào?
- Độ dài tối đa cho định danh có thể là bao nhiêu?