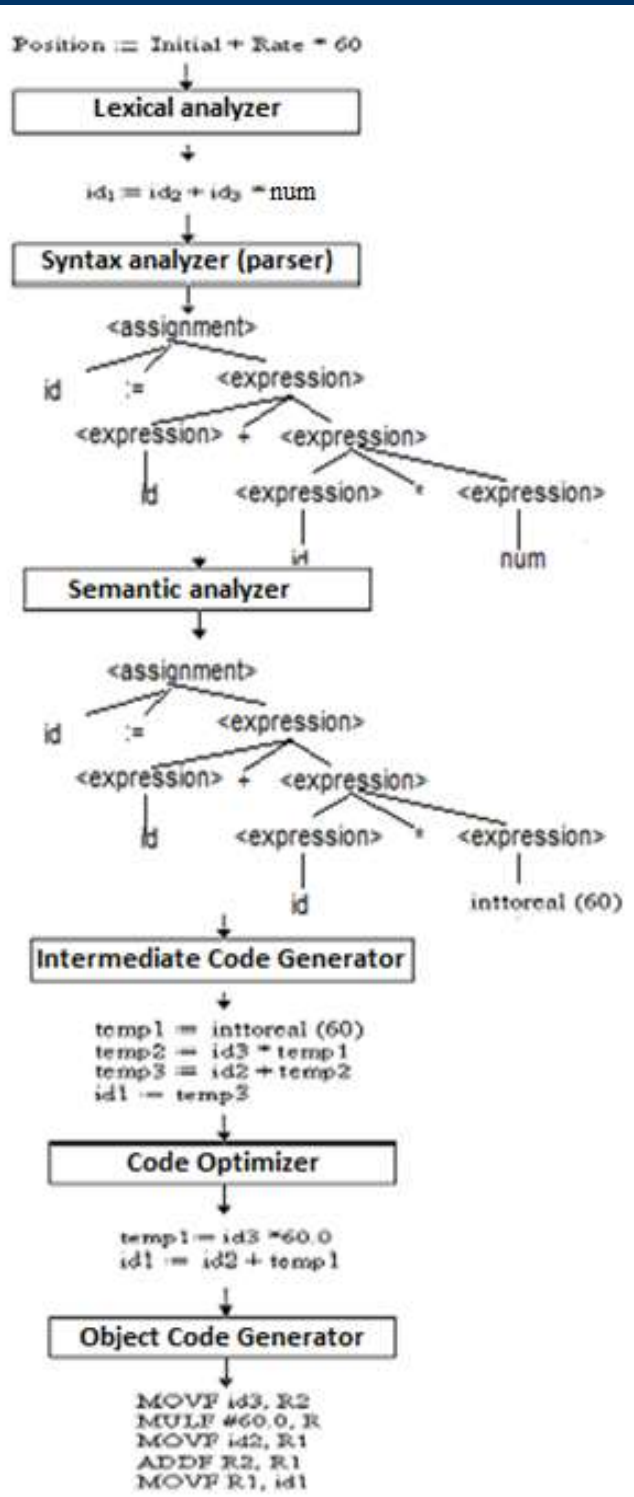
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

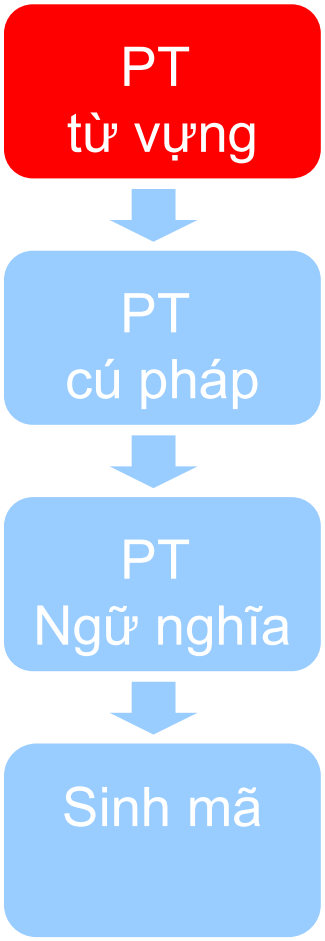
Bài thực hành
IT3323
Xây dựng chương trình dịch

Bài 1: Phân tích từ vựng

Quá trình dịch một câu lệnh gán

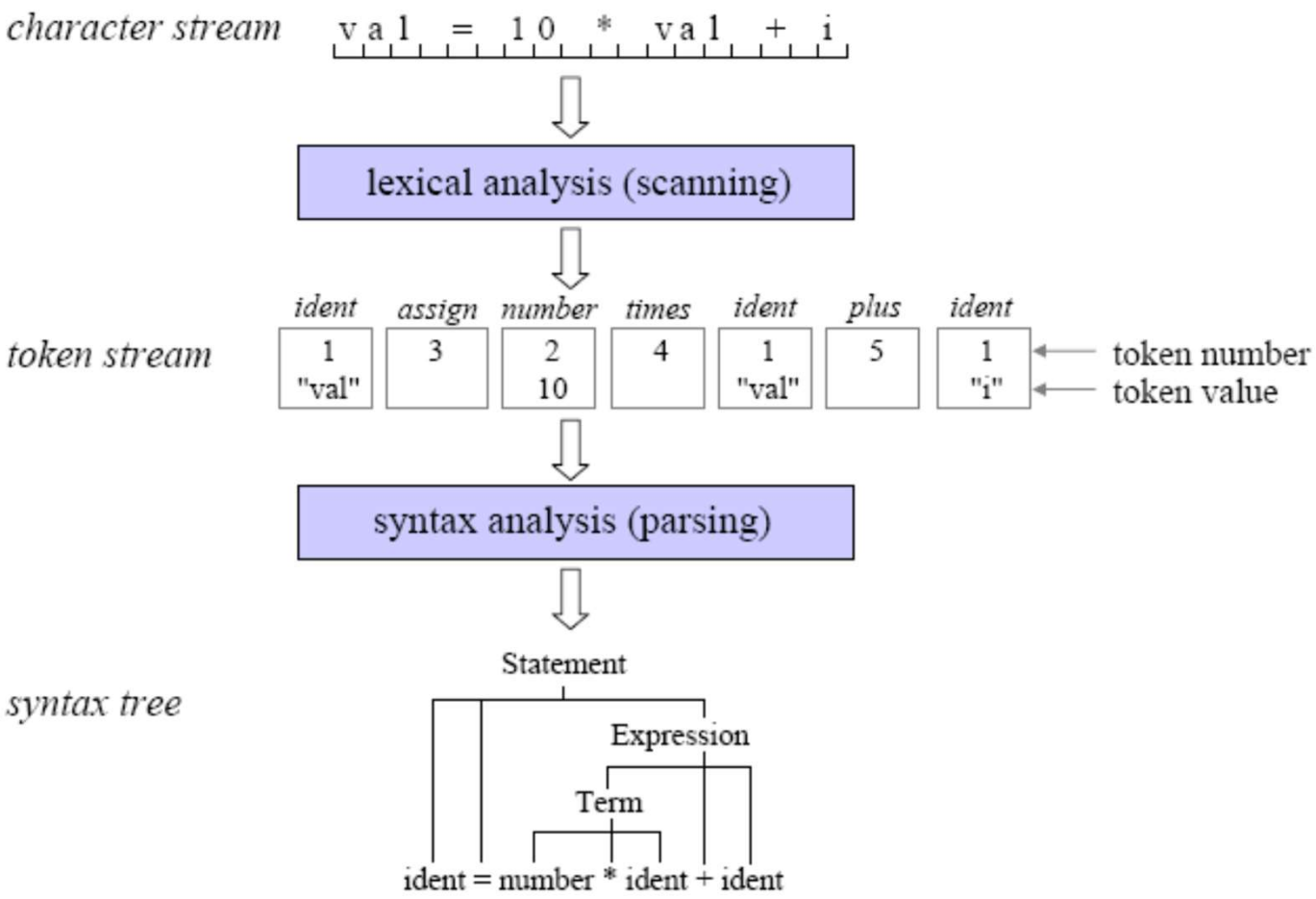


Bộ phân tích từ vựng (scanner) là gì?

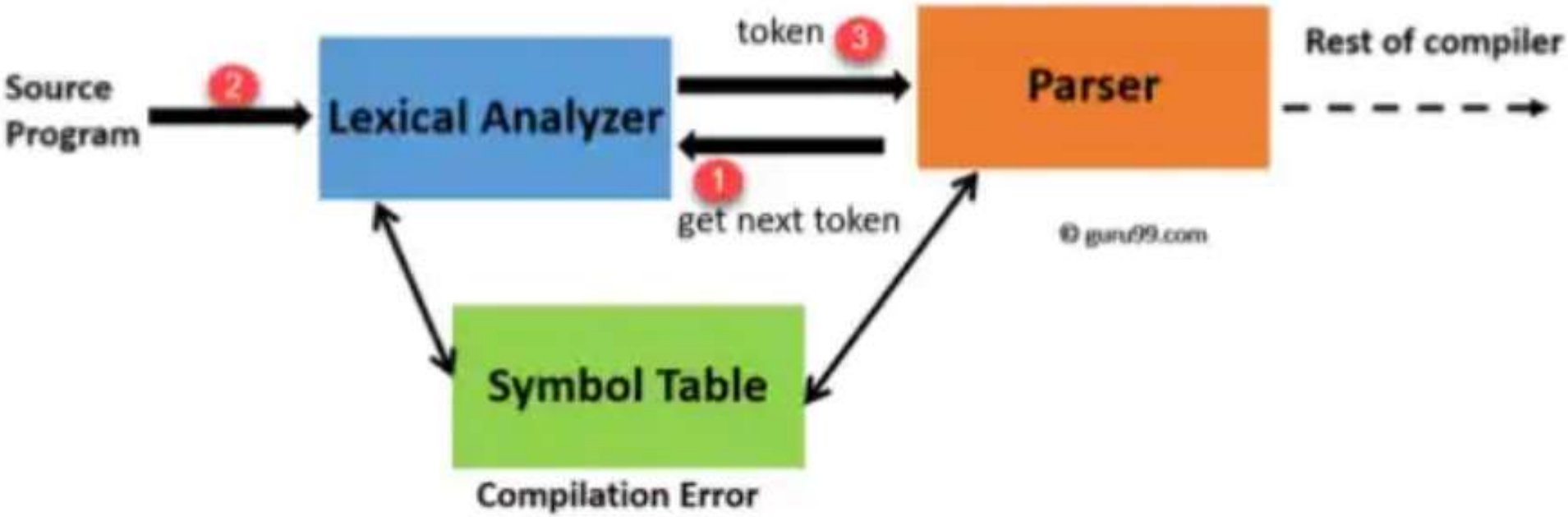


- Trong một chương trình dịch, thành phần thực hiện chức năng phân tích từ vựng gọi là *scanner*.
- Đây là pha đầu tiên trong compiler một giai đoạn

Bộ phân tích từ vựng (scanner) là gì?



Tương tác giữa bộ PT từ vựng và bộ PT cú pháp



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
  - ...
- Chuyển lần lượt các từ tố được phát hiện cho bộ phân tích cú pháp

Luật từ vựng của KPL

- Chỉ cho phép số nguyên không dấu, giá trị không vượt quá hằng INT_MAX (limits.h)
- Định danh bao gồm chữ cái (hoa + thường), chữ số, bắt đầu bằng chữ cái, độ dài không vượt quá 15. Phân biệt chữ hoa, chữ thường
- Từ khóa không phân biệt chữ hoa, chữ thường
- Chỉ cho phép hằng ký tự, là một ký tự in được, bao trong cặp nháy đơn
- Ngôn ngữ không sử dụng hằng xâu ký tự
- Dấu – chỉ được dung cho phép trừ, không phải phép đổi dấu
- Phép so sánh khác biểu thị bằng !=

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 từ tố (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 greaterness), < (comparison of lessness),

  >= (comparison of greaterness or equality), <= (comparison of lessness or equality)

Các từ tố 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

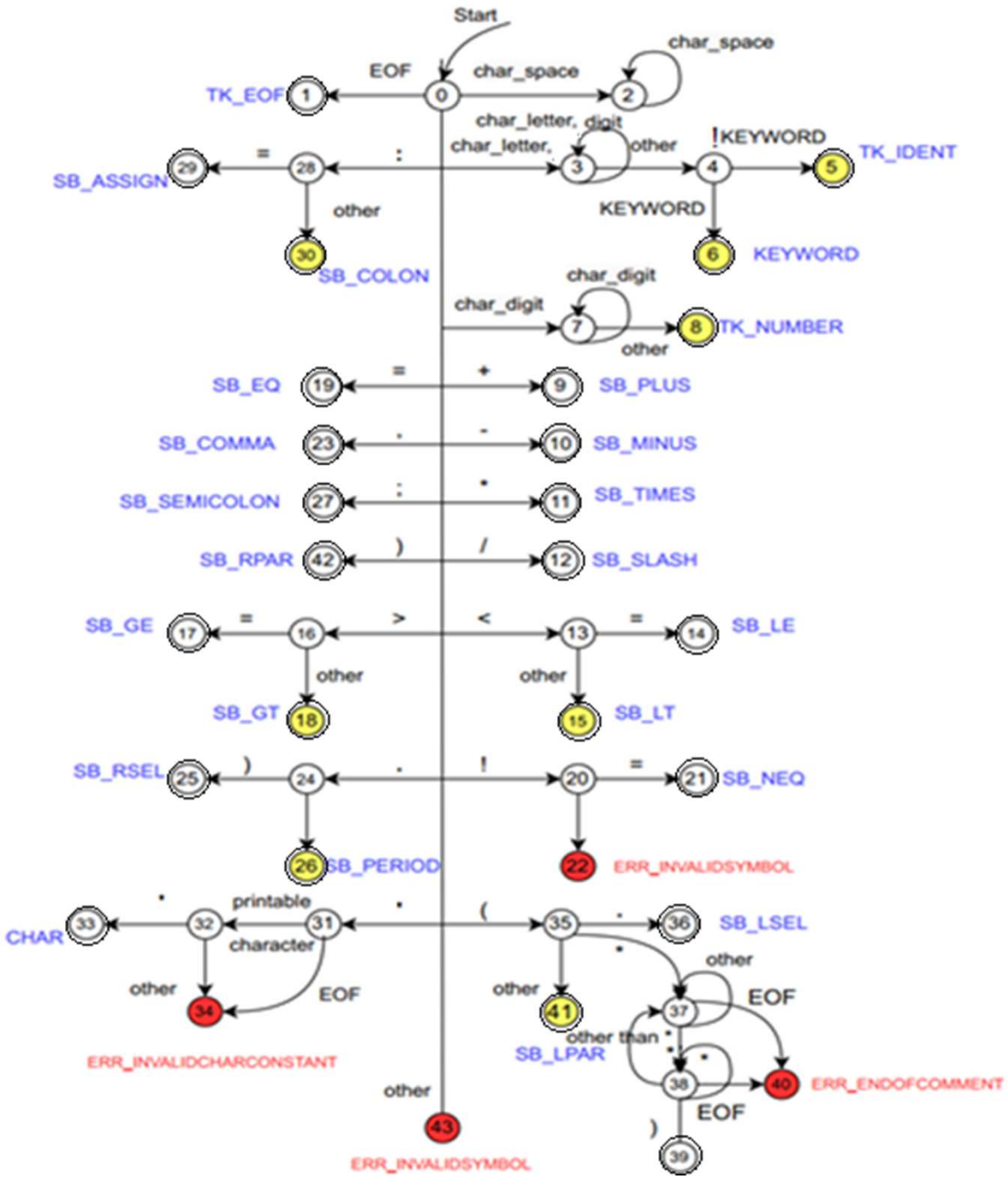
- Ngoài ra

  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 automat hữu hạn đơn định
- Bộ phân tích từ vựng (scanner) là một automat hữu hạn đơn định

Ôtômat để nhận dạng các token của KPL



- Mỗi khi hoàn tất nhận dạng một token, automat sẽ chuyển lại về trạng thái 0
- Khi có lỗi xảy ra (gặp ký tự ngoài bảng chữ cái,…), automat 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 của project

STT	Tên tệp	Nội dung
1	Makefile	Project
2	scanner.c	Tệp 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

- 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 – 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_EXCLAIMATION,    // '!'
  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 – 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 xâu 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_IDENTTOOLONG,
  ERR_NUMBERTOOLONG,
  ERR_INVALIDCHARCONSTANT,
  ERR_INVALIDSYMBOL
} ErrorCode;

// Các thông báo lỗi
#define ERM_ENDOFCOMMENT "End of comment expected!"
#define ERM_IDENTTOOLONG "Identification too long!"
#define ERM_NUMBERTOOLONG "Value of integer number exceeds the
range!"
#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

```
//Phát hiện từng token trong chương trình nguồn, sau
//khi trả về một token, quay lại trạng thái 0
int scan(char *fileName) {
  Token *token;

  if (openInputStream(fileName) == IO_ERROR)
    return IO_ERROR;
  token = getToken();
  while (token->tokenType != TK_EOF) {
      printToken(token);
    free(token);
    state =0;
    token = getToken();
     }

  free(token);
  closeInputStream();
  return IO_SUCCESS;
}
```

Xây dựng scanner – hàm xử lý chính

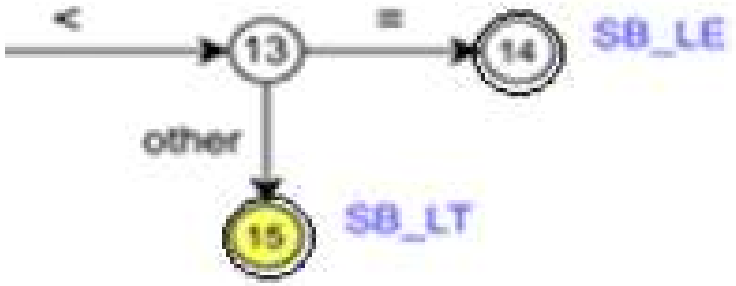
```
// Đọc một token tính từ vị trí hiện tại
Token* getToken(void)
{
  Token *token;
  switch(state)
  {
  case 0:
        if (currentChar == EOF) state =1;
        else
        switch (charCodes[currentChar])
        {
                case CHAR_SPACE:
                        state =2;break;
                case CHAR_LETTER:
                        ln=lineNo;
                        cn=colNo;
                        state =3;
                        break;
                case CHAR_DIGIT:
                        state =7;
                        break;
                case CHAR_PLUS:
                        state =12;
                        break;
```

getToken: Từ tố chỉ gồm 1 ký tự

```
case 0:
...
case CHAR_PLUS:
state = 9;
break;
...
case 9:
readChar();
return makeToken(SB_PLUS, lineNo, colNo-1);
```

getToken: Từ tố gồm 2 ký tự

```
case 13:
  readChar();
  if (charCodes[currentChar] == CHAR_EQ)
    state = 14;
  else state =15;
  return getToken();
case 14:
  readChar();
  return makeToken(SB_LE, lineNo, colNo-1);
case 15:
  return makeToken(SB_LT, lineNo, colNo-1);
```



getToken: Định danh và từ khóa

- Dùng hàm checkKeyword. Khi hàm này return TK_NONE nghĩa là xâu cần kiểm tra là định danh
- Có thể không dùng các trạng thái 5 và 6 mà return từ trạng thái 4
- Chú ý danh mục từ khóa dung chữ hoa
- Làm sao để từ khóa không phân biệt chữ hoa/chữ thường?

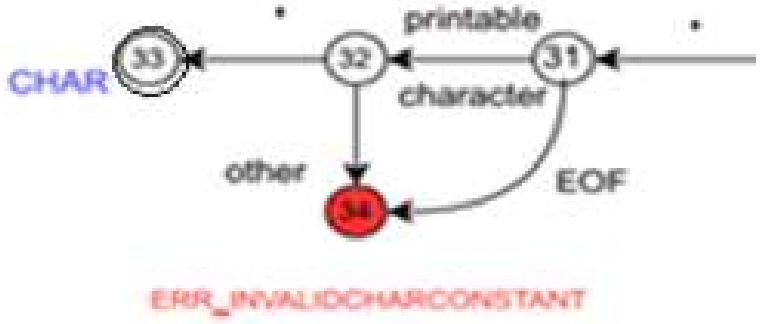
getToken: đoán nhận số

```
typedef struct {
  char string[MAX_IDENT_LEN + 1];
  int lineNo, colNo;
  TokenType tokenType;
  int value;
} Token;
```

- Chuyển xâu ký tự thành số bằng hàm atoi(stdlib.h)
- Giới hạn số: $0 \div 2^{31}-1$
  - Không có quá 10 chữ số (sau khi bỏ các chữ số 0 bên trái)
  - Nếu có đúng 10 chữ số thì về xâu ký tự là nhỏ hơn xâu chuyển đổi từ INT_MAX

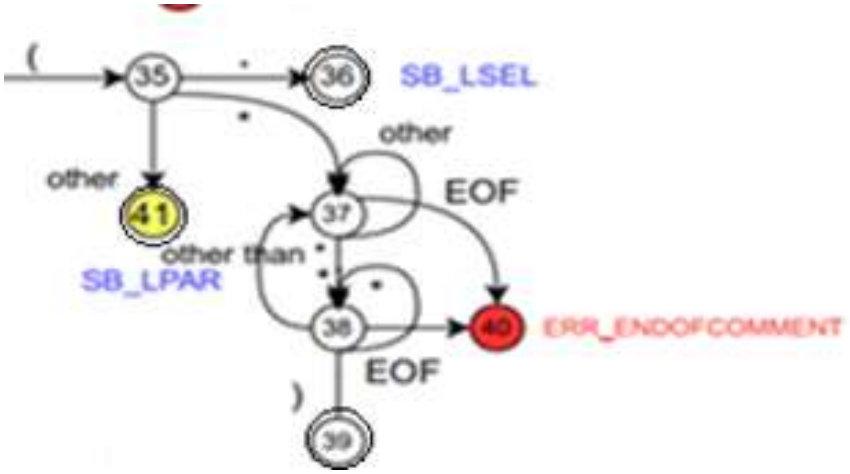
getToken: nhận biết hằng ký tự

```
case 31:
     readChar();
     if (currentChar == EOF) state=34;
     else
      if(isprint(currentChar)) state =32;
      else state =34;
     return getToken();
case 32:
     c = currentChar;
     readChar();
     if (charCodes[currentChar] == CHAR_SINGLEQUOTE)state=33;
     else state =34;
     return getToken();
case 33:
      token = makeToken(TK_CHAR, lineNo, colNo-1);
      token->string[0] =c;
      token->string[1] ='\0';
      readChar();
      return token;
case 34:
      error(ERR_INVALIDCHARCONSTANT, lineNo, colNo-2);
```



getToken: Bỏ qua chú thích

```
case 35://đọc dấu (, nhận ra SB_LSEL,SB_LPAR hoặc bỏ
        //qua chú thích
    ln = lineNo;
    cn = colNo;
    readChar();
    if (currentChar == EOF)
        state=41;//SB_LPAR
    else
     switch (charCodes[currentChar
        {
        case CHAR_PERIOD://SB_LSEL
                state =36;
            break;
        case CHAR_TIMES://Comment
            state =38;
            break;
            default:state =41; //SB_LPAR
          }
    return getToken();
```



Nhiệm vụ

- Hoàn thiện các hàm sau trong `scanner.c`
  - **`Token* getToken(void);`**