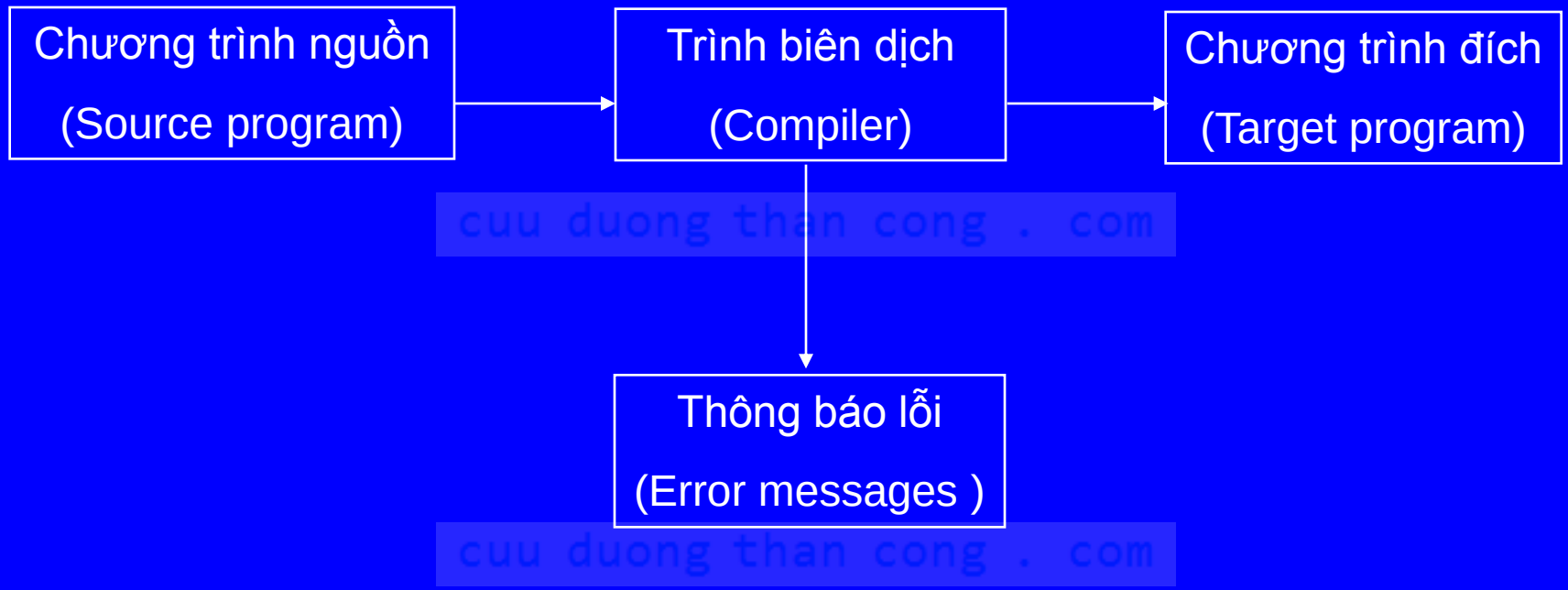
CHƯƠNG 1
Giới thiệu về chương trình dịch

Mục tiêu: Giới thiệu các khái niệm cơ bản, các giai đoạn chính khi biên dịch chương trình

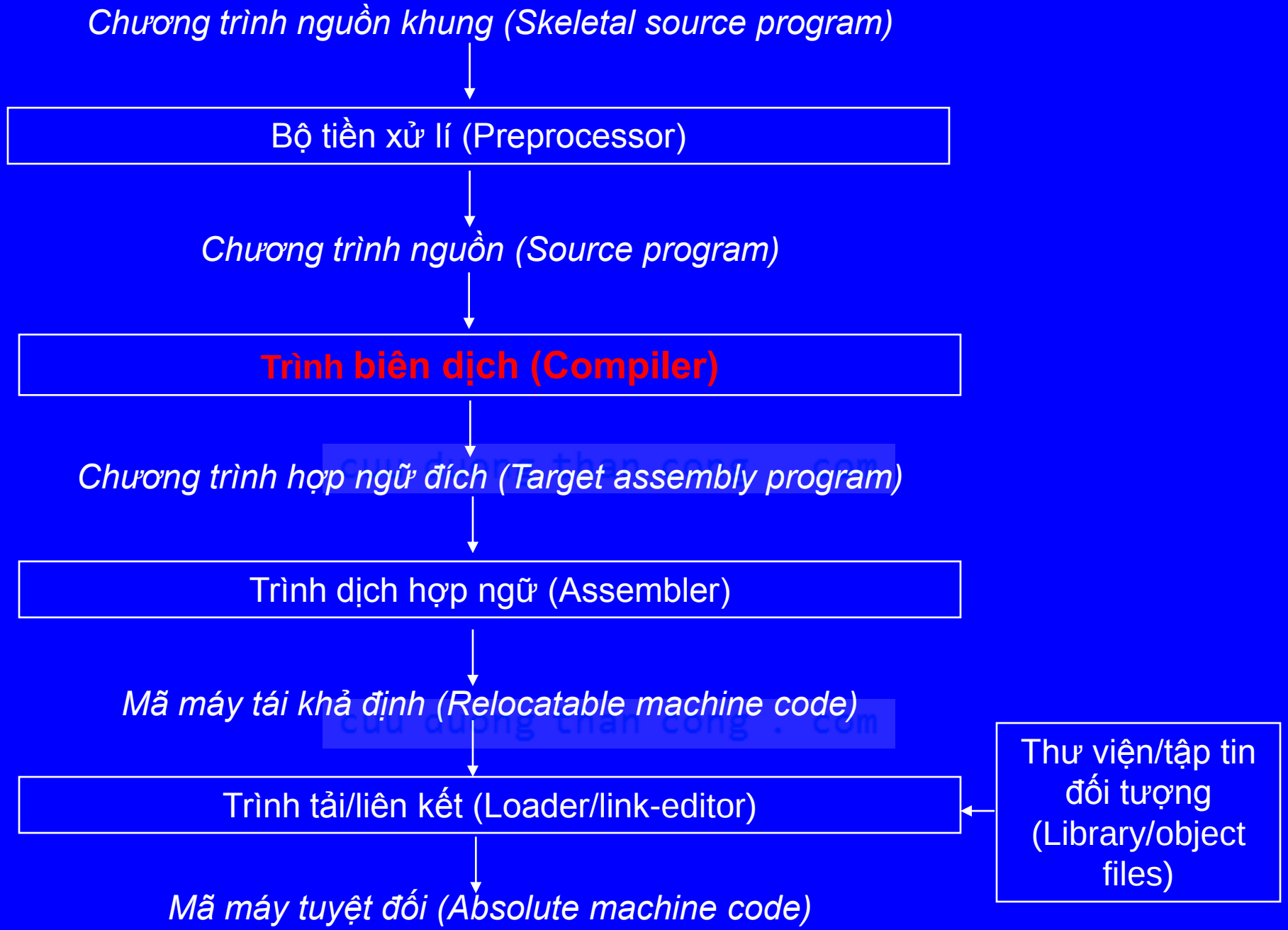
Khái niệm chương trình dịch

- Chương trình dịch (compiler) là một chương trình làm nhiệm vụ đọc một chương trình được viết bằng một ngôn ngữ - ngôn ngữ nguồn (source language) - rồi dịch nó thành một chương trình tương đương ở một ngôn ngữ khác - ngôn ngữ đích (target languague).
- Chương trình dịch ta còn gọi là trình biên dịch
- Một phần quan trọng trong quá trình dịch là ghi nhận lại các lỗi có trong chương trình nguồn để thông báo lại cho người viết chương trình



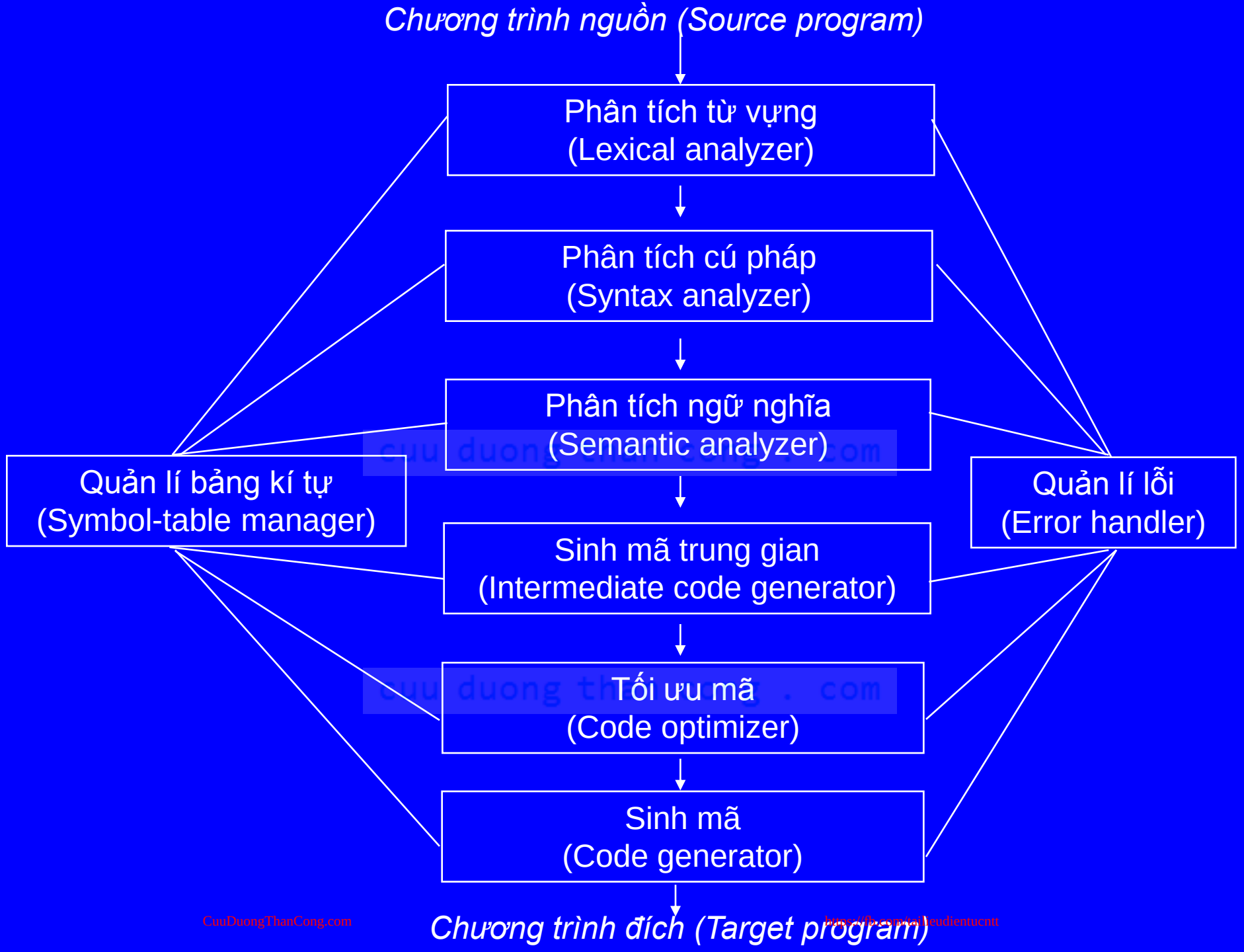
Ngữ cảnh của một trình biên dịch

- Để tạo tra một chương trình đích có khả năng thực thi (excutable) thì ngoài trình biên dịch ta phải có thêm một số chương trình khác nữa.
- Sơ đồ sau mô tả ngữ cảnh của một trình biên dịch trong một hệ thống xử lí ngôn ngữ (language-processing system)



Các giai đoạn biên dịch chương trình

- Quá trình biên dịch được chia thành nhiều giai đoạn
- Qua mỗi giai đoạn chương trình nguồn được chuyển đổi từ dạng biểu này sang một dạng biểu diễn khác
- Trong thục tế xây dựng trình biên dịch, đôi khi các giai đoạn này được nhóm lại với nhau
- Các giai doạn biên dịch được minh hoạ trong hình vẽ dưới đây



1. Phân tích từ vựng (Lexical Analysis)

- Giai đoạn phân tích từ vựng sẽ đọc chương trình nguồn từ trái sang phải (linear analysis/scanning) để tách ra thành các mã thông báo (token)
- Trong quá trình phân tích từ vựng các khoảng trắng (blank) sẽ bị bỏ qua.

- **Ví dụ :** Quá trình phân tích từ vựng cho câu lệnh gán **position := initial + rate * 60** sẽ tách thành các token như sau:
  1. Định danh (identifier) position
  2. Ký hiệu phép gán :=
  3. Định danh initial
  4. Ký hiệu phép cộng +
  5. Định danh rate
  6. Ký hiệu phép nhân *
  7. Số 60

2. Phân tích cú pháp (Syntax Analysis)

- Giai đoạn phân tích cú pháp thực hiện công việc nhóm các token của chương trình nguồn thành các cụm từ văn phạm (grammatical phrase)
- Thông thường các cụm từ văn phạm này được biểu diễn bằng cây phân tích cú pháp (parse tree) với :

  - Ngôn ngữ được định nghĩa bởi các luật sinh (production)

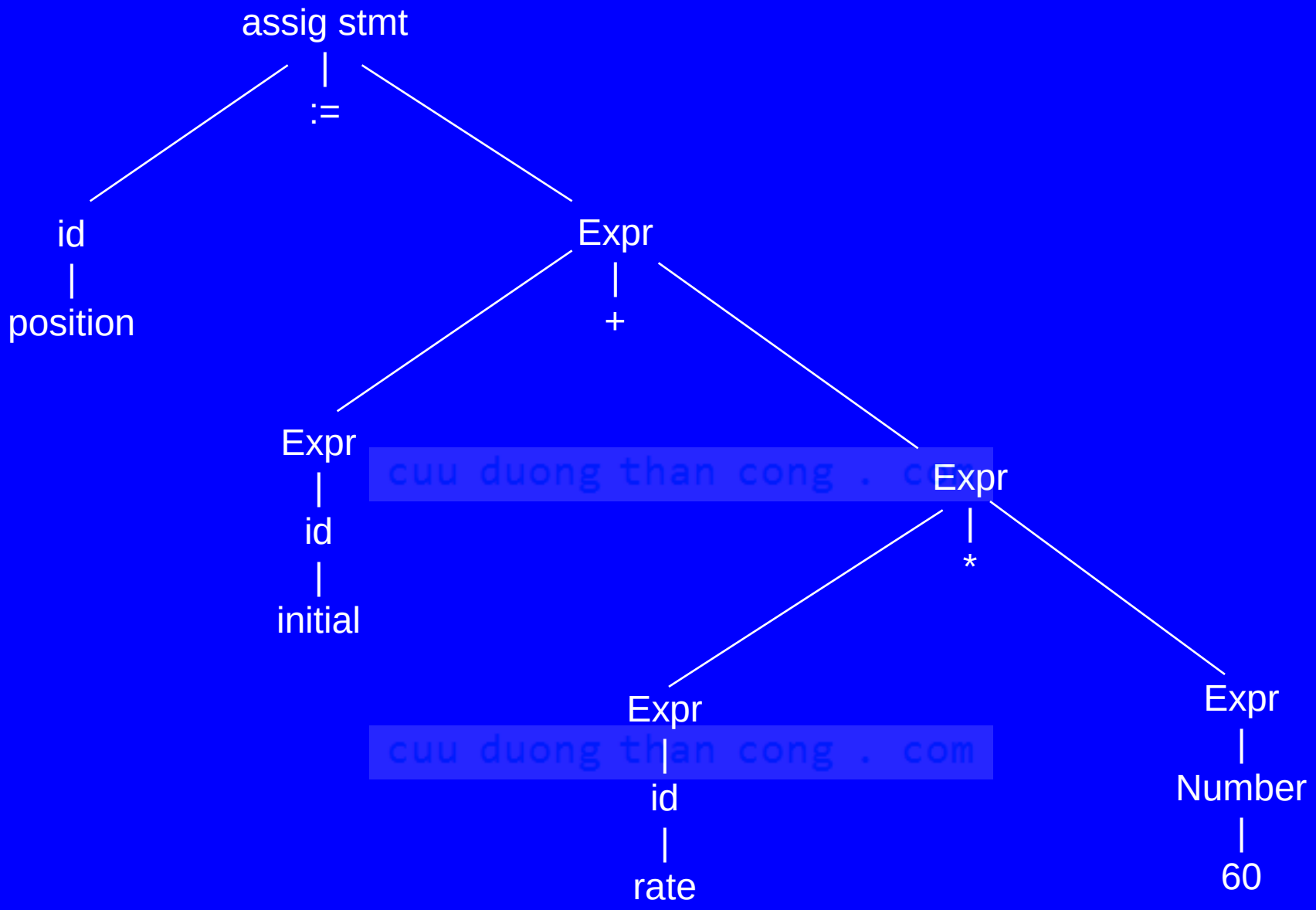
  - Phân tích cú pháp dựa vào luật sinh để xây dựng cây phân tích cú pháp.

- Ví dụ: Giả sử ngôn ngữ định nghĩa câu lệnh gán bởi các luật sinh sau :

  assig stmt $\rightarrow$ id := expr

  expr $\rightarrow$ expr + expr | expr * expr | (expr) | id | number

- Lệnh gán position := initial + rate * 60 được biểu diễn bằng parse tree như sau:



position := initial + rate * 60

- Cấu trúc phân cấp của một chương trình thường được diễn tả bởi quy luật đệ qui

Ví dụ: Định nghĩa đệ qui một biểu thức số học như sau

1) Định danh (identifier/id) là một biểu thức (expression/expr)
2) Số (number) là một biểu thức.
3) Nếu expr1 và expr2 là các biểu thức thì:

    expr1 + expr2

    expr1 * expr2

    (expr1)

cũng là những biểu thức.

Ví dụ: Nhiều ngôn ngữ lập trình định nghĩa các câu lệnh (statement) theo cách như sau :

1) Nếu id1 là một định danh và expr2 là một biểu thức thì id1 := expr2 là một lệnh (stmt).

2) Nếu expr1 là một biểu thức và stmt2 là một lệnh thì

while (expr1) do stmt2

If (expr1) then stmt2

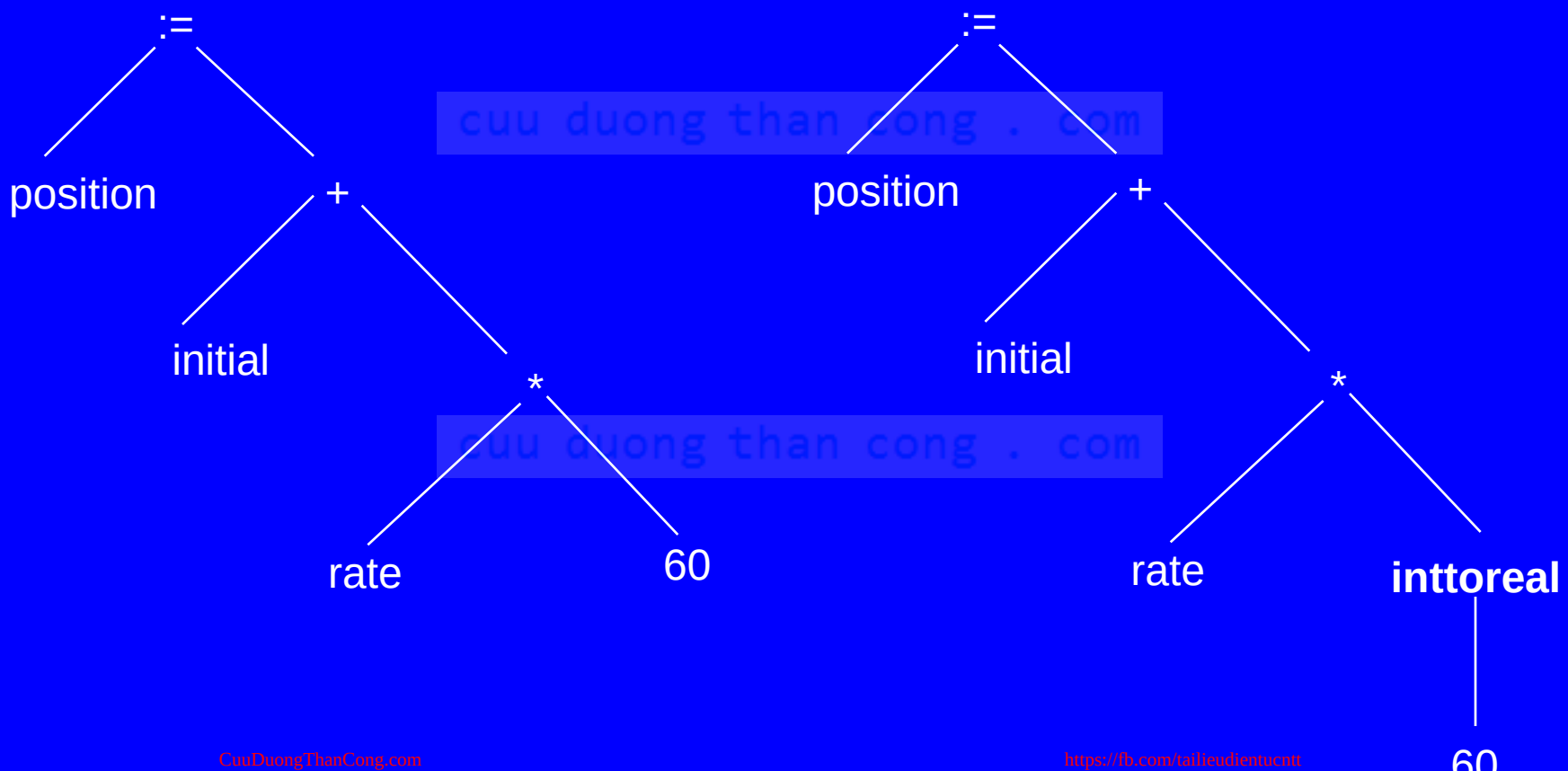
đều là các lệnh.

3. Phân tích ngữ nghĩa (Semantic Analysis)

- Giai đoạn phân tích ngữ nghĩa sẽ thực hiện việc kiểm tra xem chương trình nguồn có chứa lỗi về ngữ nghĩa hay không
- Tập hợp thông tin về các kiểu dữ liệu cho giai đoạn sinh mã về sau
- Sử dụng cấu trúc phân cấp của giai đoạn phân tích cú pháp để xác định các toán tử, toán hạng của các biểu thức và câu lệnh
- Một phần quan trọng trong giai đoạn phân tích ngữ nghĩa là kiểm tra kiểu (type checking) và ép chuyển đổi kiểu

Ví dụ: Trong câu lệnh position := initial + rate * 60

Giả sử các biến rate, initial và position được khai báo là real, 60 là số integer vì vậy trình biên dịch sẽ đổi số nguyên 60 thành số thực 60.0 bằng hàm **inttoreal**



4. Sinh mã trung gian (Intermediate Code Generator)

- Sau khi phân tích cấu trúc và ngữ nghĩa, một số trình biên dịch sẽ tạo ra một dạng biểu diễn trung gian của chương trình nguồn
- Mã trung gian có 2 đặc tính quan trọng: Dễ tạo và dễ dàng chuyển đổi thành chương trình đích
- Mã trung gian có cách biểu diễn. Thông thường người ta sử dụng dạng "mã máy 3 địa chỉ" (three-address code), tương tự như dạng hợp ngữ cho một máy mà trong đó mỗi vị trí bộ nhớ có thể đóng vai trò như một thanh ghi.

Ví dụ: Lệnh position := initial + rate * 60 khi chuyển sang dạng mã trung gian three-address code có dạng:

temp1 := inttoreal (60)

temp2 := id3 * temp1

temp3 := id2 + temp2

id1 := temp3

- Ta dùng id1, id2, id3 thay thế cho position, initial và rate để nhấn mạnh rằng tên của một nhận dạng sẽ bị thay đổi khi ta biên dịch chương trình

5. Tối ưu mã (Code Optimizer)

- Giai đoạn tối ưu mã cố gắng tối ưu mã trung gian để thu được mã máy thực hiện nhanh hơn

Ví dụ: Mã trung gian nêu trên có thể tối ưu thành:

```
temp1 := id3 * 60.0
id1 := id2 + temp1
```

6. Sinh mã (Code generation)

- Giai đoạn cuối cùng của biên dịch là sinh mã đích, thường là mã máy hoặc mã hợp ngữ.
- Vị trí các vùng nhớ gán cho các biến được chương trình sử dụng.
- Sau đó, các lệnh trung gian được dịch lần lượt thành chuỗi các chỉ thị mã máy. Vấn đề quyết định là việc gán các biến cho các thanh ghi.

Ví dụ: Sử dụng các thanh ghi (chẳng hạn R1, R2) cho việc sinh mã đích như sau:

```
MOVF    id3, R2
MULF    #60.0, R2
MOVF    id2, R1
ADDF    R2, R1
MOVF    R1, id1
```

Quản lí bảng kí tự

- Một nhiệm vụ quan trọng của trình biên dịch là ghi lại các định danh được sử dụng trong chương trình nguồn và thu thập các thông tin về các *thuộc tính* khác nhau của mỗi định danh
- Các thuộc tính cung cấp thông tin về vị trí bộ nhớ được cấp phát cho một định danh, kiểu và phạm vi của định danh
- Nếu định danh là tên của một thủ tục thì thuộc tính là các thông tin về số lượng và kiểu của các đối số, phương pháp truyền đối số và kiểu trả về của thủ tục (nếu có)

- Bảng ký hiệu (symbol table) là một cấu trúc dữ liệu mà mỗi phần tử là một bản ghi dùng để lưu trữ một định danh, các trường là các thuộc tính của định danh
- Trong quá trình phân tích từ vựng, tên các định danh được tìm thấy và nó được đưa vào bảng ký hiệu nhưng thường thì các thuộc tính của nó chưa xác định được trong giai đoạn này
- Các thuộc tính khác của các định danh sẽ được bổ sung trong các giai đoạn biên dịch tiếp theo

Ví dụ: Trong Pascal câu lệnh khai báo biến

Var position, initial, rate: real;

Sau khi phân tích từ vựng bảng quản lí kí tự có dạng như sau:

SYMBOL TABLE

1	position	.....
2	initial	.....
3	rate	.....
4		

Xử lí lỗi

- Mỗi giai đoạn biên dịch có thể gặp nhiều lỗi, ví dụ:

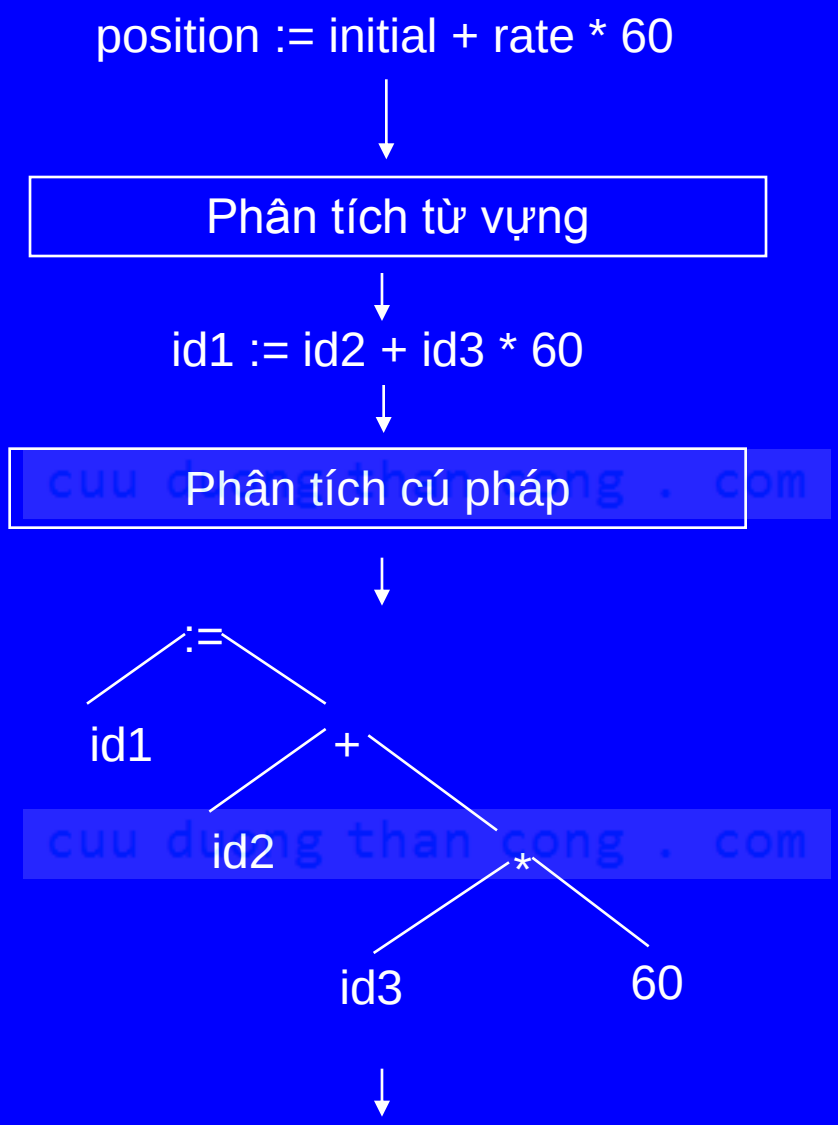
✓ Giai đoạn phân tích từ vựng gặp lỗi khi các ký tự không thể ghép thành một token.

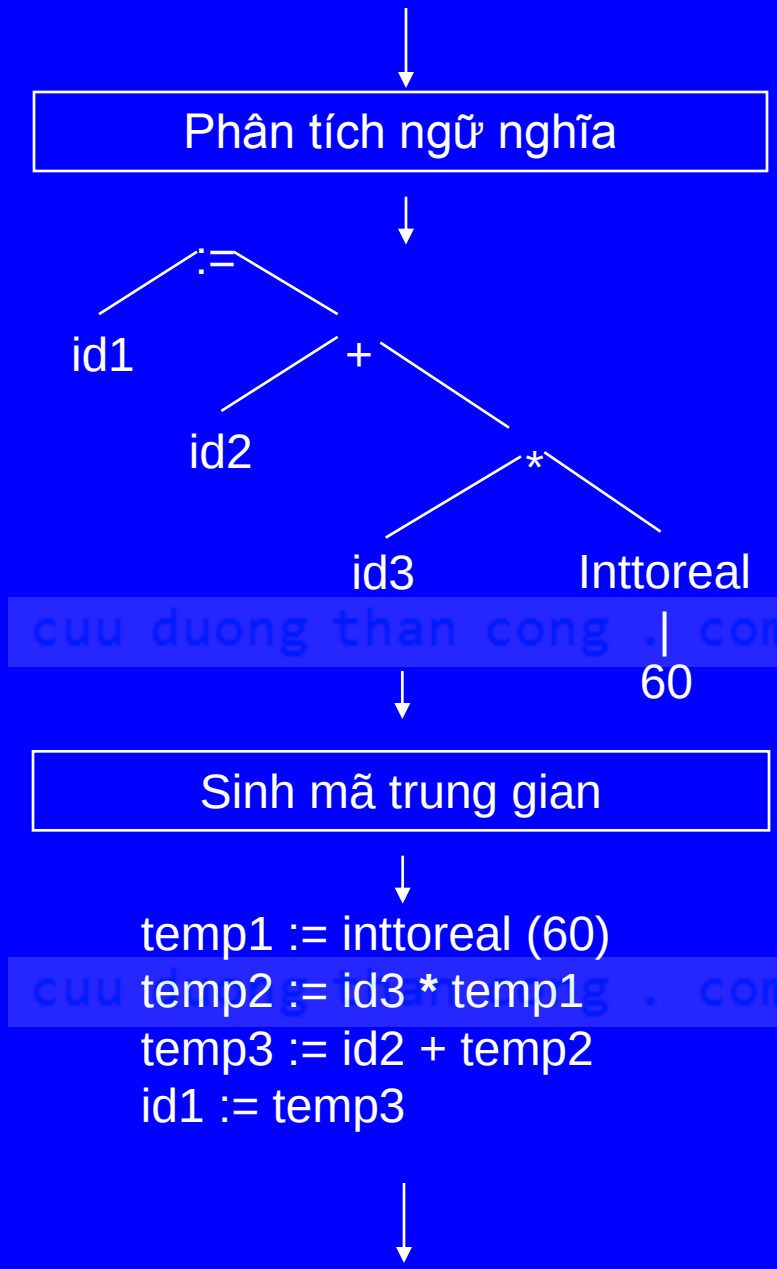
✓ Giai đoạn phân tích cú pháp gặp lỗi khi các token không thể kết hợp với nhau theo đúng cấu trúc ngôn ngữ

✓ Giai đoạn phân tích ngữ nghĩa gặp lỗi khi các toán hạng có kiểu không đúng yêu cầu của phép toán

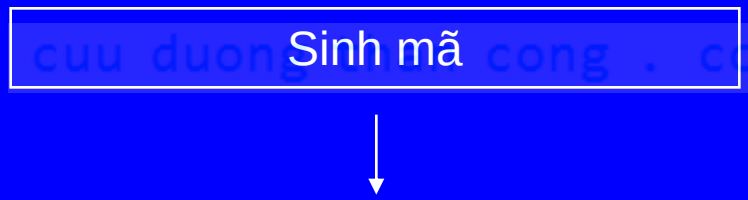
- Sau khi phát hiện ra lỗi, tùy thuộc vào trình biên dịch mà có các cách xử lý lỗi khác nhau: Quá trình biên dịch có thể dừng lại hoặc tiếp tục

Ví dụ: Biên dịch 1 câu lệnh gán





temp1 := id3 * 60.0
id1 := id2 + temp1




```
MOVF    id3, R2
MULF    #60.0, R2
MOVF    id2, R1
ADDF    R2, R1
MOVF    R1, id1
```