ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

XÂY DỰNG TRÌNH PHÂN TÍCH CÚ PHÁP CHO NGÔN NGỮ LẬP TRÌNH PYTHON

Sinh viên thực hiện:		
STT	Họ tên	MSSV
1	Cao Đức Trí	19520305
2	Huỳnh Ngọc Công Danh	19521322

TP. HỒ CHÍ MINH – 06/2021

MỤC LỤC

1. GIỚI THIỆU VỀ TRÌNH PHÂN TÍCH CÚ PHÁP

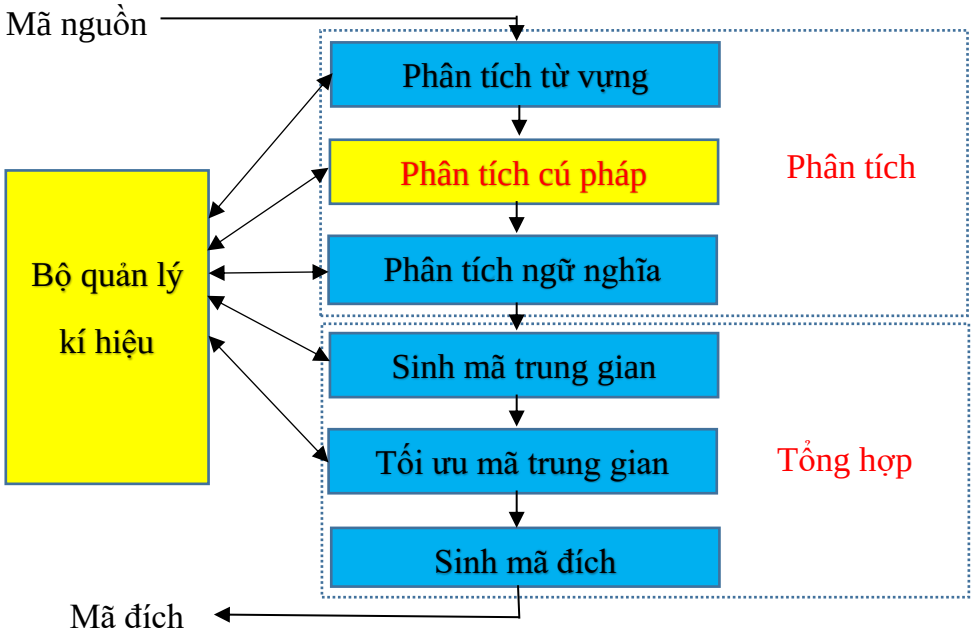
Phân tích cú pháp là một quá trình phân tích một chuỗi các biểu tượng, sử dụng trong ngôn ngữ tự nhiên, ngôn ngữ máy tính và các cấu trúc dữ liệu, tuân theo các quy tắc của bộ văn phạm đầu vào. Trong Khoa học máy tính, trình phân tích cú pháp được dùng để phân tích ngôn ngữ máy tính, mã đầu vào sẽ được phân tích thành các thành phần nhằm tạo điều kiện thuận lợi cho việc viết trình biên dịch và trình thông dịch.

2. NỘI DUNG

2.1. Khái niệm văn phạm:

- Văn phạm G là một hệ thống (Σ, Δ, P, Q) trong đó:
  - Σ là tập hữu hạn các kí hiệu kết thúc (terminal)
  - Δ là tập hữu hạn các kí hiệu không kết thúc (nonterminal)
    - Còn gọi là kí hiệu trung gian hay biến
    - $\Sigma \cap \Delta = \emptyset$
  - $S \in \Delta$ gọi là kí hiệu khởi đầu (initial)
  - P là tập hữu hạn các cặp chuỗi (α, β) được gọi là luật văn phạm hay luật sinh
    - Thường được viết là $\alpha \rightarrow \beta$
    - Chuỗi α phải có ít nhất một kí hiệu không kết thúc
- Văn phạm phi ngữ cảnh giới hạn các luật sinh phải có dạng $A \rightarrow \alpha$ trong đó $A \in \Delta$ (nói một cách dễ hiểu là vế trái của luật chỉ có 1 kí hiệu), thường sử dụng trong việc biểu diễn và phân tích cú pháp
- Văn phạm G gọi là văn phạm có đệ quy trái nếu chứa các luật dạng $A \rightarrow A\ \alpha \mid \beta$. Kí hiệu trung gian A suy dẫn ra chính nó đôi lúc gây ra khó khan trong việc sinh cây phân tích (thuật toán Top – down).
- Một văn phạm bị gọi là nhập nhằng nếu tồn tại chuỗi w có ít nhất hai cây phân tích tạo ra nó
  - Ta có thể xử lý nhập nhằng bằng cách thêm vào một số kí hiệu kết thúc phụ và một vài luật văn phạm trung gian
  - Ví dụ: $S \rightarrow S + S \mid S * S \mid$ number. Ta khử nhập nhằng bằng cách thêm vào biến trung gian
    - $S \rightarrow S + T \mid T$
    - $T \rightarrow T *$ number | number

2.2. Cấu trúc của một trình dịch:

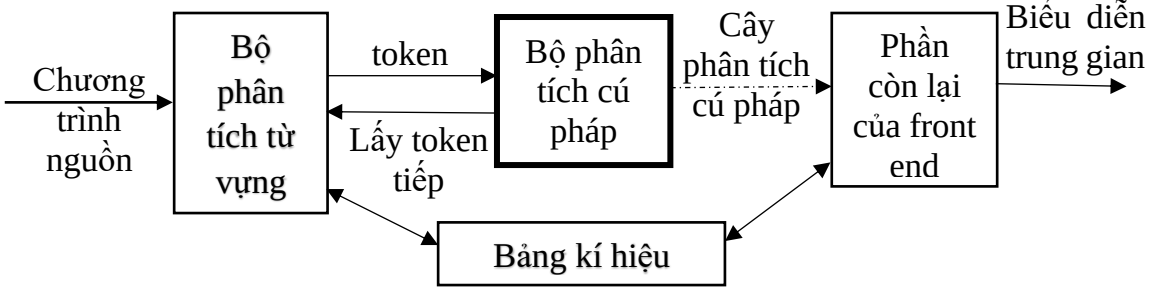


- Một trình dịch gồm hai phần chính là phân tích và tổng hợp, trong đó phần phân tích bao gồm bộ phân tích từ vựng, phân tích cú pháp và phân tích ngữ nghĩa. Phần tổng hợp gồm ba giai đoạn là sinh mã trung gian, tối ưu mã trung gian và sinh mã đích.
- Các giai đoạn trong một trình dịch:
    - Phân tích từ vựng: Xác định các đơn vị từ vựng trong mã nguồn, phân loại các đơn vị từ vựng trong các lớp như hằng, từ khóa riêng và nhập chúng vào các bảng khác nhau. Nó sẽ bỏ qua các comment trong chương trình nguồn, xác định mã thông báo không phải là một phần của ngôn ngữ
    - Phân tích cú pháp: Lấy mã thông báo từ trình phân tích từ vựng, kiểm tra xem biểu thức có đúng về mặt cú pháp hay không, báo cáo tất cả các lỗi cú pháp và xây dựng một cấu trúc phân cấp được gọi là cây phân tích cú pháp
    - Lưu trữ thông tin kiểu dữ liệu được thu thập và lưu nó trong bảng kí hiệu hoặc cây cú pháp, thực hiện kiểm tra kiểu dữ liệu. Trong trường hợp kiểu dữ liệu không khớp, một lỗi ngữ nghĩa được đưa ra.
    - Tạo mã trung gian: Mã trung gian nằm giữa ngôn ngữ cấp cao và ngôn ngữ máy, được tạo ra từ ngữ nghĩa của chương trình nguồn

- Tối ưu mã trung gian: Loại bỏ dòng mã không cần thiết và sắp xếp chuỗi các câu lệnh để tăng tốc độ thực hiện chương trình mà không lãng phí tài nguyên
- Sinh mã đích: Nhận đầu vào từ giai đoạn tối ưu hóa mã và tạo ra mã hợp ngữ hoặc mã đối tượng. Mục tiêu của giai đoạn này là phân bố lưu trữ và tạo mã máy.

2.3. Vai trò của bộ phân tích cú pháp:

- Phân tích cú pháp là giai đoạn thứ hai của trình dịch
- Nhận chuỗi các token từ bộ phân tích từ vựng, dựa theo các luật văn phạm của ngôn ngữ, xây dựng cây cú pháp cho chuỗi vào
  - Phân tích cú pháp làm việc chặt chẽ với phân tích từ vựng và thường có thể bắt đầu thực hiện công việc ngay khi phân tích từ vựng mới có những kết quả ban đầu, không cần phải đợi phân tích từ vựng kết thúc
- Phân tích cú pháp cung cấp dữ liệu cho bộ phân tích ngữ nghĩa
  - Làm việc độc lập với bộ phân tích ngữ nghĩa, chỉ trả về kết quả cho phân tích ngữ nghĩa sau khi đã hoàn thành đầy đủ hoặc tương đối đầy đủ việc tạo cây cú pháp
- Cung cấp dữ liệu về lỗi và gợi ý sửa lỗi cho bộ soạn thảo



2.4. Nhiệm vụ của bộ phân tích cú pháp:

- Phân tích cú pháp đảm nhận nhiệm vụ phức tạp nhất của trình dịch, đó là kiểm tra lỗi cú pháp của chuỗi vào
- Các nhiệm vụ chính:
  - Xây dựng cây cú pháp cho chuỗi vào
  - Thực hiện một số thao tác ngữ nghĩa phục vụ cho việc phân tích tiếp theo
    - Tập hợp thông tin kiểu dữ liệu và kiểm tra tính tương thích của kiểu dữ liệu
  - Phát hiện các lỗi về văn phạm và lựa chọn phương pháp xử lý phù hợp
    - Xử lý lỗi để tiếp tục thực hiện việc phân tích
    - Đưa ra các gợi ý sửa lỗi cho mã nguồn

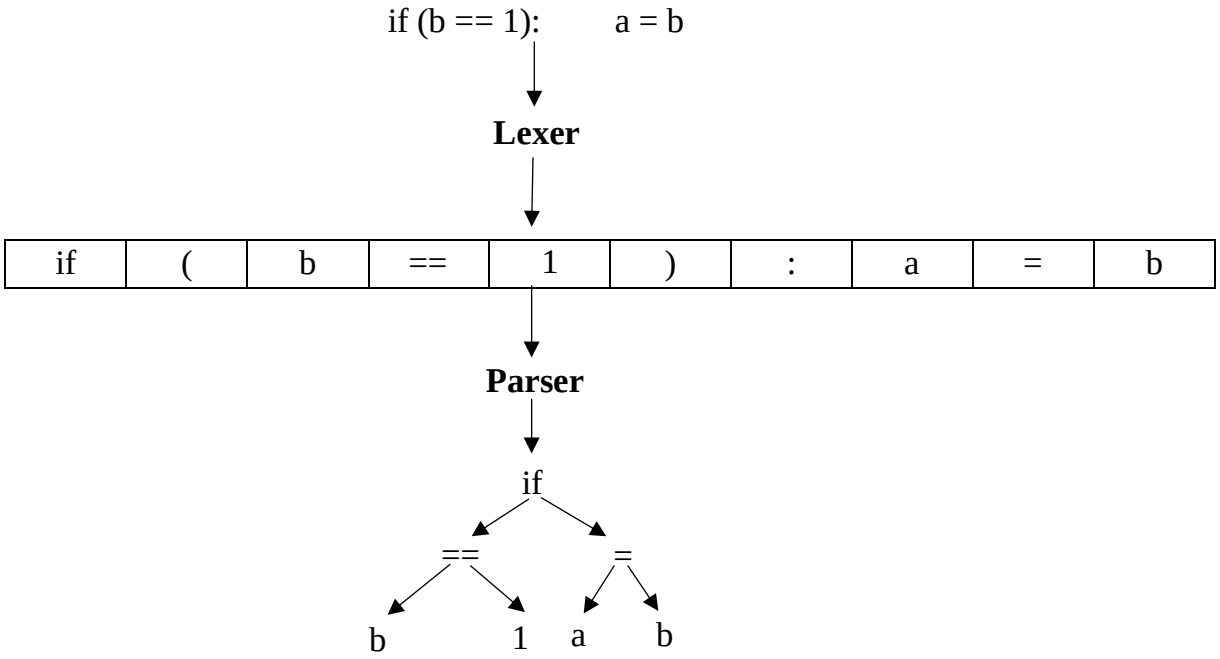
2.5. Các mục tiêu của bộ phân tích cú pháp:

- Chính xác: Kết quả phân tích cần trả về chính xác cây phân tích
- Tốc độ: Khó xây dựng các bộ phân tích cú pháp tuyến tính theo độ dài của chuỗi vào (ngoại trừ ngôn ngữ đầu vào có văn phạm quá đơn giản), nhưng bộ phân tích cú pháp cần hoạt động đủ nhanh vì vậy nên xây dựng bộ phân tích cú pháp cận tuyến tính
- Chịu lỗi: Bộ phân tích cú pháp cần có khả năng chịu lỗi và có chiến lược khắc phục lỗi phù hợp
- Hiệu quả về bộ nhớ: Bộ phân tích cú pháp cần sử dụng bộ nhớ một cách hiệu quả do việc phải lưu trữ toàn bộ cây phân tích cho mã nguồn

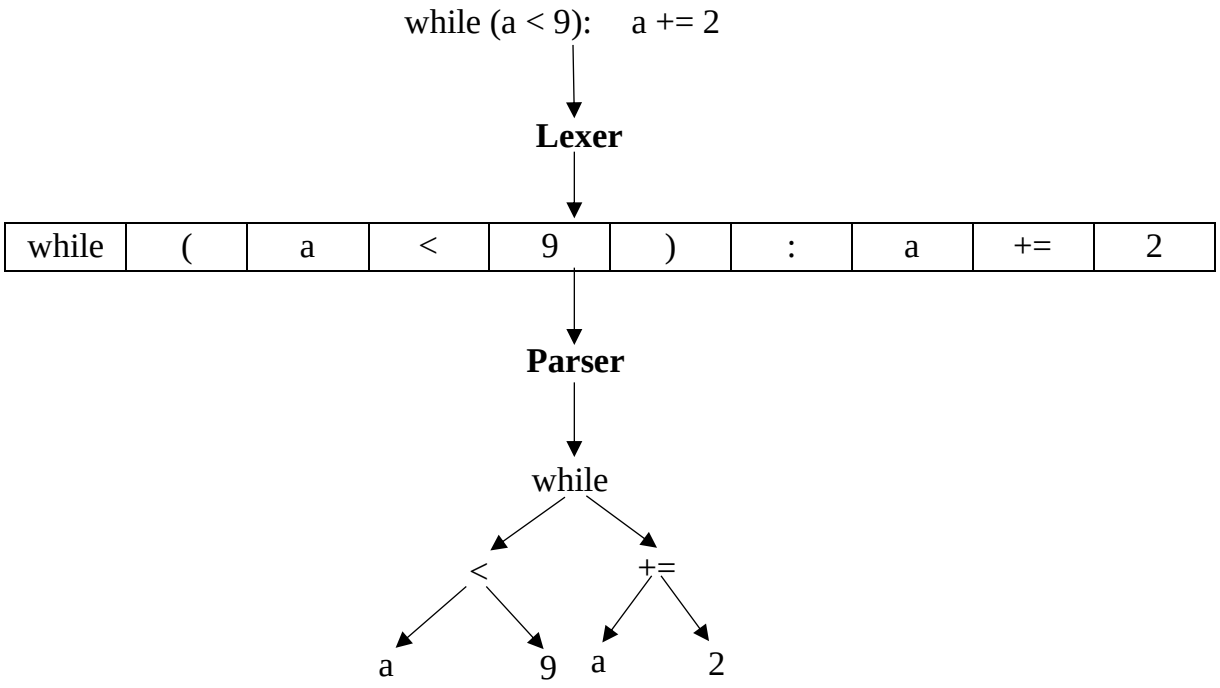
2.6. Đầu vào và đầu ra của bộ phân tích cú pháp:

2.6.1. Cây phân tích cú pháp:

- Đối với các cấu trúc cây dùng trong khoa học máy tính, cây phân tích cú pháp là một cây có gốc và thứ tự dùng để thể hiện cấu trúc cú pháp của một chuỗi theo như một vài dạng của bộ văn phạm đầu vào, trong cây phân tích gồm có:
    - Nút bên trong: Lưu trữ thông tin toán tử
    - Nút lá: Lưu trữ thông tin toán hạng
    - Đảm bảo rằng các thành phần của chương trình khớp với nhau một cách có ý nghĩa
    - Kiểm tra toán hạng được cho phép bởi ngôn ngữ nguồn
- Ví dụ: Phân tích cú pháp của các đoạn mã sau
    - **if (b == 1): a = b**

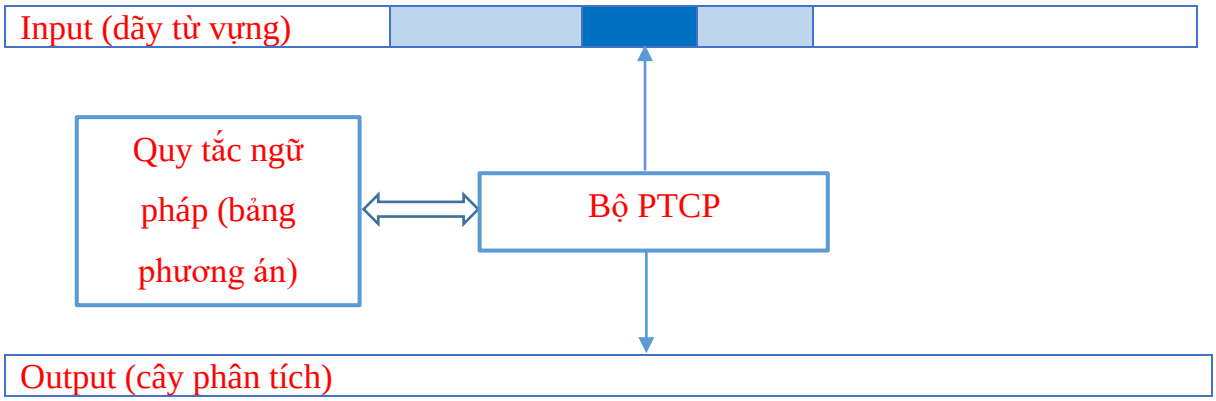

if (b == 1): a = b
Lexer
if (b == 1) : a = b
Parser
if
==
=
b
1
a
b

- **while (a < 9): a += 2**


while (a < 9): a += 2
Lexer
while (a < 9) : a += 2
Parser
while
<
+=
a
9
a
2

2.6.2. *Đầu vào của bộ phân tích cú pháp:*

- Đầu vào của bộ phân tích cú pháp là dãy các từ vựng đã được xác định chi tiết về từ loại
- Bộ phân tích cú pháp thường cần quan sát nhiều hơn một kí hiệu đầu vào để đưa ra quyết định dựng cây phân tích



2.6.3. *Đầu ra của bộ phân tích cú pháp:*

- Đầu ra của bộ phân tích cú pháp là đầu vào của bộ phân tích ngữ nghĩa, thường thì chỉ có thể hiểu đúng ngữ nghĩa khi đã xác định đầy đủ cấu trúc của câu, vì thế bộ phân tích cú pháp thường trả về cây phân tích đầy đủ cho bộ phân tích ngữ nghĩa
- Cây phân tích thường có các thành phần sau:
  - Cấu trúc cây (có nhiều lựa chọn kiểu cấu trúc dữ liệu)
  - Cấu trúc một nút cây:
    - Kí hiệu ở nút hiện tại
    - Từ vựng liên quan
    - Danh sách các nút con
    - Thông tin bổ sung, phục vụ cho việc phân tích tiếp theo

2.7. Các bước xây dựng bộ phân tích cú pháp:

- Mô tả các luật văn phạm của ngôn ngữ nguồn
  - Các mô tả này ban đầu có thể ở dạng ngôn ngữ tự nhiên
  - Đặc tả ý nghĩa các kí hiệu không kết thúc
  - Chuyển thành các luật văn phạm ở dạng chặt chẽ

- Phân tích bộ văn phạm để lựa chọn phương pháp phân tích cú pháp phù hợp nhất
    - Văn phạm có nhập nhằng hay không?
    - Văn phạm có đệ quy trái hay không?
    - Văn phạm có sự mơ hồ hay không?
    - Văn phạm có độ phức tạp ở mức độ nào?
- Lựa chọn phương pháp phân tích cú pháp phù hợp
- Lựa chọn cách xử lý trong tình huống lỗi cú pháp, sinh các gợi ý sửa lỗi và các tình huống cần phải tổ hợp ngữ nghĩa

3. CÁC THƯ VIỆN HỖ TRỢ VÀ THUẬT TOÁN TOP – DOWN

3.1. Các thư viện hỗ trợ:

- Nltk hay Natural Language Toolkit – Bộ công cụ ngôn ngữ tự nhiên, là một thư viện được viết bằng python hỗ trợ xử lý ngôn ngữ tự nhiên. Bằng cách cung cấp các cơ chế và kĩ thuật xử lý ngôn ngữ phổ biến, nó giúp cho việc xử lý ngôn ngữ tự nhiên trở nên dễ dàng và nhanh chóng hơn.
- Qt5 là thư viện hỗ trợ cho việc thiết kế ra các giao diện giao tiếp với người dùng của các phần mềm chức năng. Hỗ trợ với Python 2.x và 3.x.

3.2. Các chiến lược phân tích cú pháp:

- Được chia thành ba nhóm:
    - Thử sai (quay lui): Top – Down, Bottom – Up (Shift – Reduce)
    - Quy hoạch động: CYK, Earley,…
    - Tất định: LL, LR,…
- Ngoài ra còn có một số phương pháp khác dựa trên đặc điểm của ngôn ngữ để áp dụng các kĩ thuật hiệu quả

3.3. Thuật toán Top – Down:

- Ý tưởng:
    - Xây dựng cây phân tích theo lối từ gốc đến lá
    - Xuất phát từ gốc là kí tự bắt đầu, cố gắng áp dụng các luật sinh trong văn phạm để xây dựng một cây có tập các nhãn của lá từ trái qua phải là xâu vào
- Các bước thực hiện:
    - Input: Văn phạm G(V_T, V_N, S, P) và một xâu vào x

- Output: Cây suy dẫn cho xâu vào x hoặc trả lời x không nhận được
- Thuật toán:
  - Bước 1: Xây dựng một cây chỉ có nút S làm gốc, gọi S là nút hoạt động và kí hiệu cần phân tích là kí hiệu đầu tiên trên xâu vào
  - Bước 2:
    - ✓ Nếu nút hoạt động là một kí hiệu không kết thúc A
      - ➢ Chọn luật sinh đầu tiên có A làm vế trái để tạo ra k con trực tiếp của A (A → $X_1X_2…X_k$), sau đó lấy X_1 làm nút hoạt động
      - ➢ Nếu k = 0 (A → ε) thì nút hoạt động là nút bên phải của A trên cây
    - ✓ Nếu nút hoạt động là một kí hiệu kết thúc a thì sẽ so sánh a với kí hiệu cần phân tích
      - ➢ Trùng nhau: Nút hoạt động là nút bên phải của a trên cây và kí hiệu cần phân tích là kí hiệu tiếp theo trên xâu vào
      - ➢ Không trùng: Quay lại một bước để thử một luật sinh khác. Nếu tất cả các luật sinh đã được thử thì quay lại bước trước đó
  - Quá trình dừng lại khi:
    - ✓ Tạo ra cây suy dẫn cho xâu vào
    - ✓ Thử hết khả năng nhưng không xây dựng được cây suy dẫn. Tức là xâu không nhận được
  - Điều kiện dừng: Văn phạm không có đệ quy trái

- Ví dụ 1: Cho văn phạm sau
  - S → cAd
  - A → ab | a

➢ Xâu vào là cad

- Đánh số luật sinh:

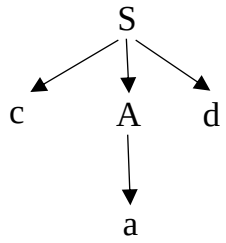
  (1) S → cAd

  (2) A → ab

  (3) A → a

Bước	Dạng câu	Phần xâu vào chưa phân tích	Luật sinh
0	S	cad	S → cAd
1	cAd	cad	So sánh
2	Ad	ad	A → ab
3	abd	ad	So sánh
4	bd	d	So sánh, quay lại bước 2
5	Ad	ad	A → a
6	ad	ad	So sánh
7	d	d	So sánh, kết thúc

- Cây phân tích cú pháp:



- Ví dụ 2: Cho văn phạm sau:
  - S → aSbS | aS | c

✓ Xâu vào là aacbc

✓ Đánh số luật sinh:

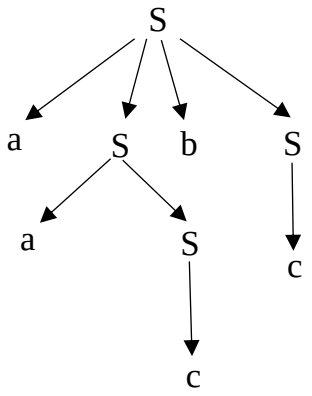
(1) S → aSbS

(2) S → aS

(3) S → c

Bước	Dạng câu	Phần xâu vào chưa phân tích	Luật sinh
0	S	aacbc	S → aSbS
1	aSbS	aacbc	So sánh
2	SbS	acbc	S → aSbS
3	aSbSbS	acbc	So sánh
4	SbSbS	cbc	S → c
5	cbSbS	cbc	So sánh
6	bSbS	bc	So sánh
7	SbS	c	S → c
8	cbS	c	So sánh
9	bS		Quay lại bước 2
10	SbS	acbc	S → aS
11	aSbS	acbc	So sánh
12	SbS	cbc	S → c
13	cbS	cbc	So sánh
14	bS	bc	So sánh
15	S	c	S → c
16	c	c	So sánh, kết thúc

✓ Cây phân tích cú pháp:



4. MÔ TẢ SẢN PHẨM

4.1. Cách hoạt động của chương trình:

- Input: Đầu vào của chương trình là một file chứa cú pháp của chương trình cần kiểm tra
- Output: Trả về kết quả sau khi kiểm tra cú pháp của python thông qua các luật đã được tạo từ trước
- Cách hoạt động của chương trình:
    - Đọc dữ liệu input chuyển đổi chúng thành một danh sách các token
    - Sử dụng các luật đã được tạo từ người lập trình để kiểm tra
    - Trả về kết quả thông qua giao diện đồ họa

4.2. Ý tưởng tạo ra chương trình:

- Chương trình sẽ gồm ba phần chính:
    - Phần Lexical
    - Phần Parser
    - Phần giao diện

4.2.1. Phần Lexical:

Dữ liệu của chương trình là một file chứa đoạn code cần kiểm tra.

- Bước 1: Chương trình sẽ đọc từng dòng của file.
- Bước 2: Một biến có kiểu dữ liệu là chuỗi sẽ chứa các kí tự có liên quan đến cú pháp của ngôn ngữ python (không tính kí tự xuống dòng). Nếu gặp các kí tự không liên quan thì sẽ đến bước 3.
    - Các kí tự liên quan đến cú pháp python: a-z, 0-9, kí tự hai chấm, kí tự tab,…
- Bước 3: Lưu biến đó như một từ tố (token) vào danh sách.
- Bước 4: Nếu file đã được đọc hết thì danh sách từ tố (token) sẽ được chuyển cho bộ phận parser để phân tích.

4.2.2. Phần Parser:

- Bước 1: Tạo ra một danh sách các luật của ngôn ngữ python bằng việc sử dụng thư viện NLTK để triển khai. Một biến syntax sẽ được tạo ra với giá trị True để kiểm tra cú pháp.
    - Ví dụ: Một số luật của chương trình.
        - input_stmt -> indentifer '=' 'input' '(' ')'
        - for_stmt -> 'for' indentifer 'in' 'range' value 'colon' 'tab' block
- Bước 2: Từ dữ liệu do phần lexical cung cấp, áp dụng các luật được triển khai từ bước 1. Sử dụng thuật toán Top - Down được thư viện NLTK lập trình sẵn để tạo ra cây phân tích cú pháp.
    - Ví dụ: rd_parser = nltk.RecursiveDescentParser(grammar1)
- Bước 3: Nếu cây cú pháp được tạo ra thì thư viện NLTK sẽ trả về cho chúng ta cây cú pháp.
    - Ví dụ:
        - len(list(rd_parser.parse(sent))) == 0
        - Nếu phép so sánh trên trả về True thì cây cú pháp không được tạo ra. Còn trả về False thì cây cú pháp sẽ được tạo ra.
- Bước 4: Từ phép so sánh ở bước 3. Nếu cây cú pháp không được tạo ra thì một lỗi sẽ được sinh ra và cú pháp trên được xác định là không chính xác. Nếu không có lỗi nào được sinh ra thì cú pháp trên được xác định là chính xác.
    - Ví dụ:

```
try:
    if len(list(rd_parser.parse(sent))) == 0:
    # Tạo ra lỗi bằng cách xóa dữ liệu của danh sách trống
        del list(rd_parser.parse(sent))[0]

except:

    # Cú pháp sẽ được xác định là không chính xác

    syntax = False
```

4.2.3. Giao diện:

- Giao diện được tạo ra bằng qt5, có chức năng cho phép chúng ta kéo thả.
- Giao diện bao gồm:
    - Khung nhập dữ liệu đầu vào (input).
    - Nút để kiểm tra dữ liệu.
    - Khung để xuất dữ liệu đầu ra (output).

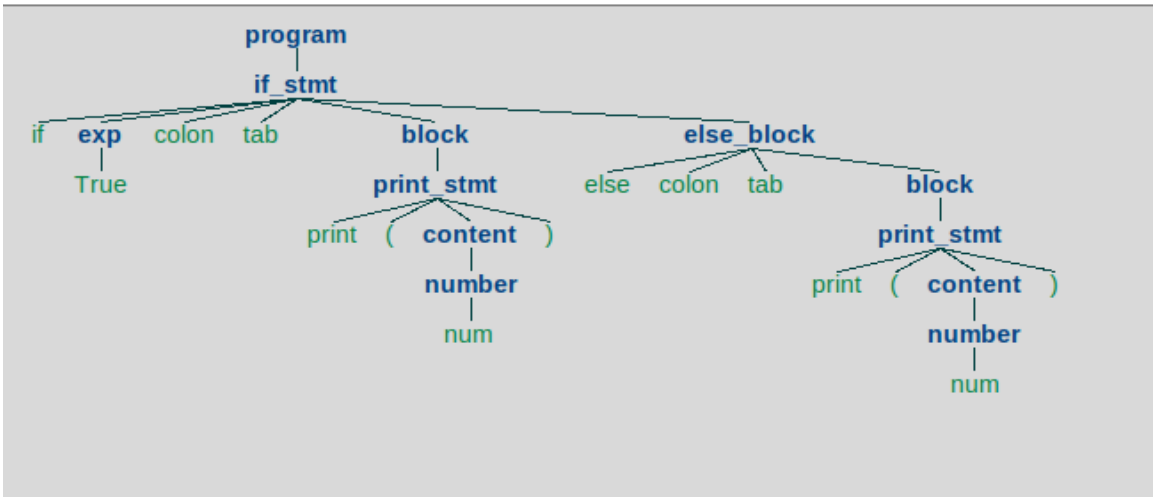
4.3. Demo chương trình:

- Ví dụ về một đoạn mã sai cú pháp trong python, khi đó cây phân tích cú pháp sẽ không được sinh ra.

Hình 1. Kết quả giao diện của chương trình khi cú pháp không chính xác.

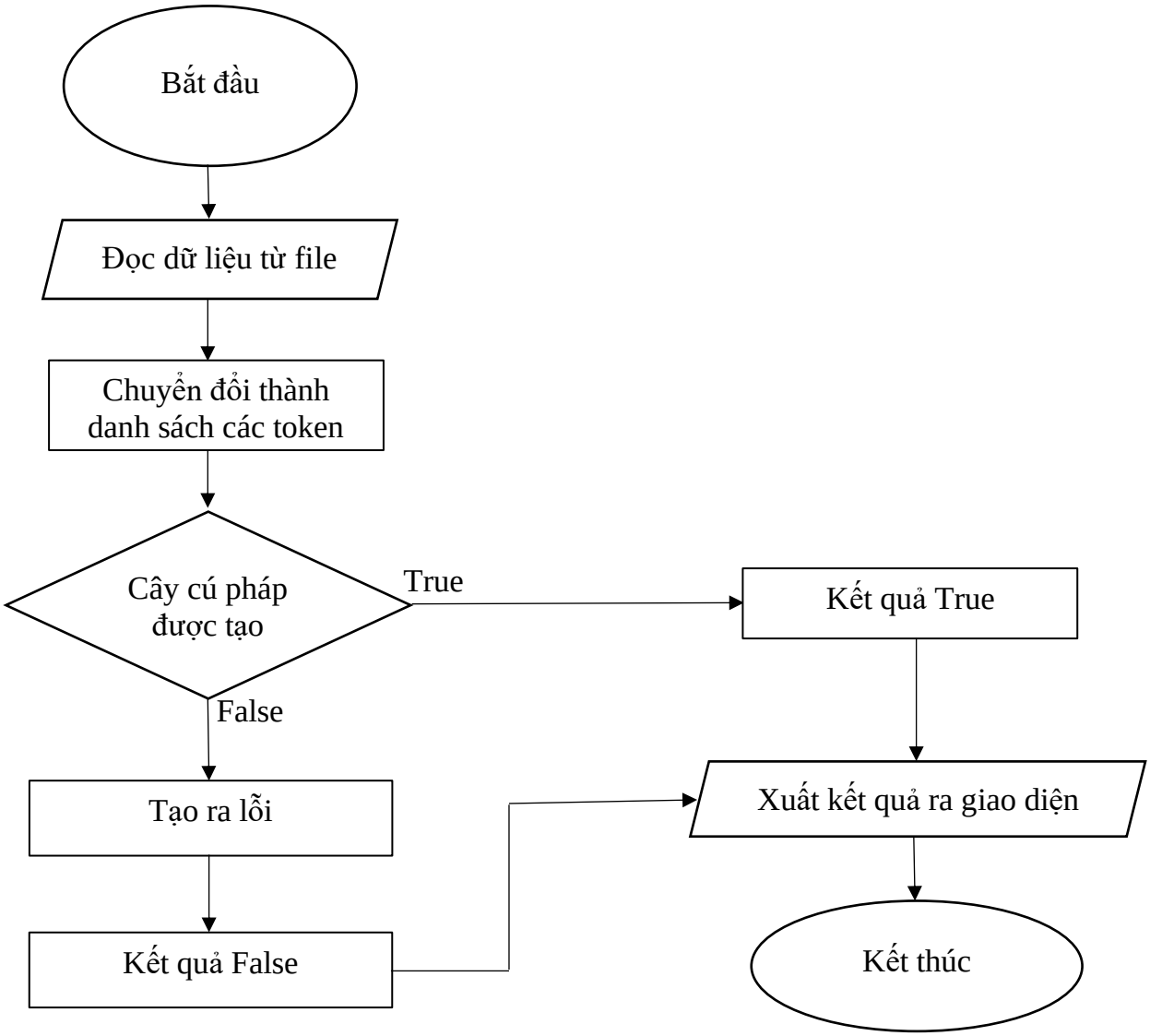
- Khi cú pháp của đoạn mã đúng, một cây phân tích cú pháp sẽ được sinh ra.

Hình 2. Kết quả giao diện của chương trình khi cú pháp chính xác.



Hình 3. Cây phân tích được sinh ra khi cú pháp chính xác.

4.4. Lưu đồ thuật toán:



4.5. Chương trình minh họa bộ phân tích cú pháp:

Link mã nguồn: https://github.com/danhhuynh25029/LexicalAndParser

5. ĐÁNH GIÁ VỀ THUẬT TOÁN TOP – DOWN VÀ CÁCH CẢI TIẾN

5.1. Đánh giá về thuật toán Top – Down:

- Thuật toán dạng thử sai quay lui, không cắt nhánh, độ phức tạp tính toán là hàm mũ.
- Không làm việc được văn phạm có đệ quy trái vì không cắt nhánh nên dẫn đến lặp vô hạn.

5.2. Cách cải tiến tăng tốc độ tính toán:

- Tập trung vào việc cài đặt cắt nhánh.
  - Cắt nhánh khi trong A có terminal không có trong w.
  - Cắt nhánh khi số terminal trong A nhiều hơn trong w.
- Tính toán trước các bước không có cơ hội về đích để loại bỏ bớt những tình huống không cần thiết.
- Sử dụng lại những kết quả đã duyệt cũ.

TÀI LIỆU THAM KHẢO

[1] Hoàng Dân Hiệp. Phân tích cú pháp trong python: Công cụ và thư viện (Phần 1). Link: https://helpex.vn/article/phan-tich-cu-phap-trong-python-cong-cu-va-thu-vien-phan-1-5c6b1a5aae03f628d053bdc3

[2] Gabriele Tomassetti. A Guide to Parsing: Algorithms and Technology (Part 8). Link: https://dzone.com/articles/a-guide-to-parsing-algorithms-and-technology-part-2

[3] Cú pháp chương trình. Link:

https://courses.uit.edu.vn/pluginfile.php?file=%2F280763%2Fmod_resource%2Fcontent%2F1%2FCu-phap-chuong-trinh.pdf

[4] Trung Nguyen. Các giai đoạn của trình biên dịch. Link:

https://comdy.vn/trinh-bien-dich/cac-giai-doan-cua-trinh-bien-dich/

[5] Cây phân tích cú pháp. Link:

https://vi.wikipedia.org/wiki/C%C3%A2y_ph%C3%A2n_t%C3%ADch_c%C3%BA_ph%C3%A1p

[6] Bài 7 : Lập trình giao diện với PyQt5 cho RaspberryPi - Phần 1. Link:

https://mlab.vn/index.php?_route_=17161-bai-7-lap-trinh-giao-dien-voi-pyqt5-cho-raspberrypi-phan-1.html

PHỤ LỤC PHÂN CÔNG NHIỆM VỤ

STT	Thành viên	Nhiệm vụ
1	Cao Đức Trí	Soạn slide, thuyết trình, viết báo cáo
2	Huỳnh Ngọc Công Danh	Viết code, thuyết trình, viết báo cáo

HẾT