**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN**

**CHƯƠNG TRÌNH DỊCH**

**Đề tài: Tính First và Follow**

**GIẢNG VIÊN HƯỚNG DẪN: THS. NGUYỄN THỊ MINH HỶ**

**SINH VIÊN THỰC HIỆN:**

1. **TRẦN VĂN SONG Mã SV: 102150298**
2. **LƯƠNG LÊ ĐẠT Mã SV: 102150265**

**LỚP: 15TCLC2**

**NHÓM: 15.15B**

**Đà Nẵng 5 - 2018**

**MỤC LỤC**

[MỞ ĐẦU 4](#_Toc515528035)

[DANH MỤC HÌNH VẼ 5](#_Toc515528036)

[GIỚI THIỆU ĐỀ TÀI 6](#_Toc515528037)

[1. Tên đề tài 6](#_Toc515528038)

[2. Lý do chọn đề tài 6](#_Toc515528039)

[3. Mục đích của đề tài 6](#_Toc515528040)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 7](#_Toc515528041)

[1.1. Ý tưởng 7](#_Toc515528042)

[1.2. Văn phạm 7](#_Toc515528043)

[1.2.1. Khái niệm về văn phạm 7](#_Toc515528044)

[1.2.2. Tính đúng đắn của văn phạm phi ngữ cảnh 7](#_Toc515528045)

[1.3. First và Follow 8](#_Toc515528046)

[1.3.1. First 8](#_Toc515528047)

[1.3.1.1. Định nghĩa First(α) 8](#_Toc515528048)

[1.3.1.2. Tính First 8](#_Toc515528049)

[1.3.2. Follow 9](#_Toc515528050)

[1.3.2.1. Định nghĩa Follow(A) 9](#_Toc515528051)

[1.3.2.2. Tính Follow 9](#_Toc515528052)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 11](#_Toc515528053)

[2.1. Phát biểu bài toán 11](#_Toc515528054)

[2.2. Cấu trúc dữ liệu 11](#_Toc515528055)

[2.3. Thuật toán 13](#_Toc515528056)

[2.3.1. Kiểm tra văn phạm: 13](#_Toc515528057)

[2.3.2. Tìm First: 20](#_Toc515528058)

[2.3.3. Tìm Follow: 23](#_Toc515528059)

[CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 26](#_Toc515528060)

[3.1. Tổ chức chương trình 26](#_Toc515528061)

[3.2. Ngôn ngữ cài đặt 26](#_Toc515528062)

[3.3. Kết quả 26](#_Toc515528063)

[3.3.1. Giao diện chính của ứng dụng 26](#_Toc515528064)

[3.3.2. Kết quả thực thi của ứng dụng 29](#_Toc515528065)

[3.3.2.1. Tính First của văn phạm G 29](#_Toc515528066)

[3.3.2.2. Tính Follow của văn phạm G 35](#_Toc515528067)

[3.4. Nhận xét 35](#_Toc515528068)

[CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 36](#_Toc515528069)

[4.1. Kết luận 36](#_Toc515528070)

[4.2. Hướng phát triển 36](#_Toc515528071)

[TÀI LIỆU THAM KHẢO 37](#_Toc515528072)

[PHỤ LỤC 37](#_Toc515528073)

# MỞ ĐẦU

Chương trình dịch là chương trình dùng để dịch một chương trình (CT nguồn) viết trên một ngôn ngữ lập trình nào đó (NN nguồn) sang một chương trình tương đương (CT đích) trên một ngôn ngữ khác (NN đích). Đối với người lập trình, Chương trình dịch là một trong những học phần rất quan trọng. Học phần này được xem như là một trong những nền tảng của lập trình máy tính. Nó giúp chúng ta hiểu được cách một ngôn ngữ lập trình được tạo ra, hoạt động như thế nào. Sau khi học xong học phần lý thuyết, nhóm chúng em đã nghiên cứu và thực hiện đồ án này như là một cách để củng cố và mở rộng kiến thức. Thông qua quá trình thực hiện đồ án, em đã nắm bắt được phần nào những kỹ thuật quan trọng trong việc xây dựng nên một ngôn ngữ lập trình, cách mà một trình biên dịch, trình thông dịch hoạt động để từ một chương trình nguồn có thể trả về chương trình đích sao cho máy tính có thể hiểu được. Bài toán “Tính first và follow” mà nhóm chúng em nghiên cứu và trình bày trong báo cáo sau đây là một ví dụ để chúng ta thấy được cách một văn phạm được tạo ra và first, follow của nó được tính như thế nào.

Chúng em xin cảm ơn cô Nguyễn Thị Minh Hỷ đã bổ sung ý tưởng và tận tình giúp chúng em thực hiện đồ án này. Chúng em cũng mong nhận được góp ý từ phía cô để bài làm của chúng em hoàn thiện hơn.

DANH MỤC HÌNH VẼ

Hình 1: Vị trí First và Follow trong một chuỗi.

Hình 2: Giao diện chính của ứng dụng.

Hình 3: Giao diện chính của ứng dụng.

Hình 4: Giao diện chính của ứng dụng.

Hình 5: Giao diện chính của ứng dụng.

Hình 6: Nhập các ký hiệu chưa kết thúc.

Hình 7: Nhập các ký hiệu kết thúc.

Hình 8: Nhập ký hiệu bắt đầu.

Hình 9: Tập các sản xuất.

Hình 10: Kiểm tra văn phạm đúng.

Hình 11: Kiểm tra văn phạm sai.

Hình 12: First(E)

Hình 13: First(E’)

Hình 14: First(T’)

Hình 15: First(F)

Hình 16: Tập các Follow của văn phạm G.

# GIỚI THIỆU ĐỀ TÀI

1. **Tên đề tài**

Tính First và Follow

* **Dữ liệu vào:** Văn phạm G.
* **Phương pháp gợi ý:** Cấu trúc lưu trữ văn phạm. Tìm hiểu giải thuật tính First và Follow trong sách Compiler của tác giả Đỗ Quang dịch.
* **Dữ liệu ra:** Tập First và Follow.

1. **Lý do chọn đề tài**

Để củng cố và rèn luyện kỹ năng lập trình, củng cố và nhớ lại các kiến thức về chuỗi (String), dach sách (List), mảng (Array), các kiến thức về văn phạm, về First và Follow của văn phạm cũng như giải thuật tính First và Follow của chúng.

1. **Mục đích của đề tài**

Giới thiệu, trình bày các khái niệm cơ bản về văn phạm, về First và Follow của văn phạm, các ứng dụng của First và Follow trong chương trình dịch như xây dựng bảng phân tích M, phục hồi lỗi theo chiến lược Panic Mode, phân tích cú pháp bằng phương pháp tiên đoán, … và những giải thuật để xây dựng nên một văn phạm cũng như tính First và Follow đã trình bày chi tiết cùng với việc phân tích và cài đặt, xây dựng ứng dụng trên máy tính.

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

* 1. **Ý tưởng**

Trong quá trình xây dựng nên một ngôn ngữ lập trình, việc phân tích cú pháp cho ngôn ngữ đó là một trong những vấn đề quan trọng, một trong những hoạt động cần thiết để xây dựng nên một bộ phân tích cú pháp là tính First và Follow của một văn phạm. Tuy nhiên đối với một văn phạm phức tạp thì việc tính toán First và Follow trở nên khó khăn. Do đó, đòi hỏi phải xây dựng một ứng dụng để tính First và Follow của văn phạm để giúp quá trình tính toán trở nên dễ dàng hơn, tiết kiệm được phần lớn thời gian trong công việc.

* 1. **Văn phạm**
     1. **Khái niệm về văn phạm**

Một chương trình là một dãy các câu lệnh, một câu lệnh là một dãy các từ, một từ được tạo ra từ các ký tự. Chương trình, câu lệnh, từ (hay còn gọi là xâu) được tạo ra theo một quy tắc nhất định, quy tắc đó chính là văn phạm. Hay nói cách khác, văn phạm là cơ chế để sản sinh ra ngôn ngữ.

Chương trình trước khi được thực thi thì phải thông qua giai đoạn biên dịch để kiểm tra xem nó có bị các lỗi từ vựng hay lỗi cú pháp không. Để việc kiểm tra được chính xác, đòi hỏi việc xây dựng văn phạm để tạo ra ngôn ngữ lập trình cũng phải chính xác.

* + 1. **Tính đúng đắn của văn phạm phi ngữ cảnh**

Văn phạm phi ngữ cảnh G được xác định thông qua 4 thành phần: Σ, ∆, S, P.

Trong đó:

* Σ là tập hữu hạn các ký hiệu kết thúc (terminal)
* Δ là tập hữu hạn các ký hiệu không kết thúc (nonterminal)

• Còn gọi là ký hiệu trung gian hay biến

• Σ ∩ Δ = ∅

* S ∈ Δ gọi là ký hiệu khởi đầu (initial)
* P: tập các sản xuất (production) có dạng A → α với A ∆; α (Σ ∆)\*; A là vế trái, α là vế phải.



Ví dụ: S → 0S| 1S| 0| 1

Kí hiệu (symbol): khái niệm cơ sở để xây dựng ngôn ngữ, không thể định nghĩa một cách hình thức.

Ví dụ: các chữ số, các chữ cái, các dấu kí hiệu,…

Văn phạm phi ngữ cảnh trên sản sinh ra ngôn ngữ là một số nhị phân.

Một văn phạm phi ngữ cảnh đúng thì mới sinh ra được ngôn ngữ. Văn phạm đó phải thỏa mãn những điều kiện sau:

* Không sử dụng dấu cách trong ký hiệu.
* Không có ký hiệu kết thúc hoặc chưa kết thúc chưa được sử dụng trong sản xuất.
* Không có sản xuất vế trái bằng vế phải.
* Không có ký hiệu vô sinh:

Ký hiệu A ∆ được gọi là ký hiệu vô sinh khi A + α ∑\*

* Không có ký hiệu không đạt đến được:

Ký hiệu A ∈ ∆ được gọi là ký hiệu không đạt đến được khi S + αAβ

* 1. **First và Follow**

Các tập First và Follow được sử dụng trong quá trình thiết kế trình biên dịch. First được sử dụng trong các trình phân tích cú pháp LL (LL parsers) là các trình phân tích cú pháp từ trên xuống để đọc hoặc diễn giải từ trái sang phải. Follow được sử dụng trong các trình phân tích cú pháp LR (LR parsers) là các trình phân tích cú pháp từ dưới lên để đọc hoặc diễn giải từ phải sang trái.

* + 1. **First**
       1. **Định nghĩa First(α)**

Giải sử α là một chuỗi các ký hiệu văn phạm, First(α) là tập hợp các ký hiệu kết thúc mà nó bắt đầu một chuỗi được suy dẫn từ α.

* Giả sử α = X1X2 … Xn
* Thêm vào First(α): First(X1) – {}
* Với mọi i = 2,3, …, n; nếu First(Xk) chứa với mọi k = 1, 2, …, i-1 thì thêm vào First(α): First(Xi) – {}
* Nếu với mọi i = 1, 2, …, n; nếu First(Xi) chứa thì thêm vào First(α)
  + - 1. **Tính First**

Tính First(X):

* Nếu X là kí hiệu kết thúc thì First(X) là {X}.
* Nếu X → là một luật sinh (production) thì thêm vào First(X).
* Nếu X → Y1Y2Y3 … Yk là một luật sinh (production) thì:

1. Thêm tất cả các ký hiệu kết thúc khác của First(Y1) vào First(X).
2. Nếu First(Y1) thì tiếp tục thêm vào First(X) tất cả các ký hiệu kết thúc khác của First(Y2).
3. Nếu First(Y1) First(Y2) thì thêm tất cả các ký hiệu kết thúc khác First(Y3).
4. Tiếp tục như vậy cho tới Yk.
5. Thêm vào First(X) nếu i=1 → k First(Yi).

Ví dụ: Xét văn phạm G

E → T E’

E’ → + T E’ |

T → F T’

T’ → \* F T’ |

F → (E) | id

* First(E) = First(T) = First(F) = {(, id}
* First(E’) = {+, }
* First(T) = {\*, }
  + 1. **Follow**
       1. **Định nghĩa Follow(A)**

Tập hợp các ký hiệu kết thúc a mà nó xuất hiện ngay sau A (bên phải của A) trong một dạng câu nào đó.

* Tức là tập hợp các ký hiệu kết thúc a sao cho tồn tại một dẫn xuất dạng S \* αAaβ.
* Chú ý rằng nếu A là ký hiệu phải nhất trong một dạng câu nào đó thì $ ∈ Follow(A) ($ là ký hiệu kết thúc chuỗi nhập).
  + - 1. **Tính Follow**

Tính Follow(A): áp dụng các quy tắc sau cho đến khi không thể thêm gì vào mọi tập Follow được nữa

* Đặt $ vào Follow(S), trong đó S là ký hiệu bắt đầu của văn phạm và $ là ký hiệu kết thúc chuỗi nhập.
* Nếu có một luật sinh A → αBβ thì thêm mọi phần tử khác của First(β) vào trong Follow(B).
* Nếu có luật sinh A → αB hoặc A → αBβ mà ∈ First(β) thì thêm tất cả các phần tử trong Follow(A) vào Follow(B).

Ví dụ: Xét văn phạm G

E → T E’

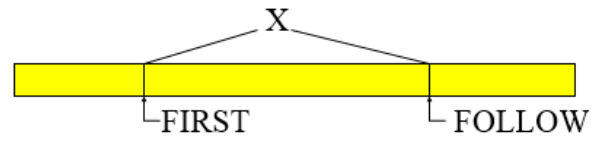
E’ → + T E’ |

T → F T’

T’ → \* F T’ |

F → (E) | id

* Follow(E) = Follow(E’) = {$, )}
* Follow(T) = Follow(T’) = {+, ), $}
* Follow(F) = {\*, +, ), $}



*Hình 1. Vị trí First và Follow trong một chuỗi*

# CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

1. **Phát biểu bài toán**

Bài toán tính First và Follow là bài toán xác định các tập First và Follow của một văn phạm phi ngữ cảnh G cho trước, với G thỏa mãn những điều kiện của một văn phạm phi ngữ cảnh.

**Dữ liệu vào:** Văn phạm phi ngữ cảnh G chứa

* Tập ký hiệu kết thúc ∑
* Tập ký hiệu chưa kết thúc ∆
* Ký hiệu bắt đầu S
* Tập các sản xuất P

**Dữ liệu ra:** Các tập First và Follow của văn phạm G nếu văn phạm G đúng

1. **Cấu trúc dữ liệu**

* Sử dụng struct để lưu trữ các sản xuất

public struct SanXuat

{

//Vế trái của sản xuất

public string vetrai;

//Vế phải của sản xuất

public string[] vephai;

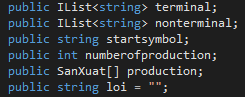
//Số kí hiệu ở vế phải

public int sokhvephai;

}

* Sử dụng lớp để lưu trữ văn phạm G và các phương thức để kiểm tra tính đúng đắn của văn phạm cũng như tính First và Follow

Ta có danh sách các biến sẽ được sử dụng chủ yếu sau đây:



Trong đó:

* Terminal: Tập các ký hiệu kết thúc.
* Non terminal: Tập các ký hiệu chưa kết thúc.
* Startsymbol: Ký hiệu bắt đầu.
* Numberofproduction: Số sản xuất.
* Production: Mảng tập hợp các sản xuất.
* Loi: Lỗi sẽ bắt được (nếu có).

Ngoài ra còn có các hàm khác như:

//Hàm kiểm tra tất cả các ký hiệu của văn phạm có sử dụng dấu cách không?

public bool kokyhieusudungdaucach()

//Hàm kiểm tra ký hiệu kết thúc s đã được sử dụng chưa?

public bool dasudungkhkt(string s)

//Hàm xác định tập ký hiệu kết thúc chưa sử dụng

public IList<string> tapkhktchuasudung()

//Hàm kiểm tra ký hiệu chưa kết thúc s đã được sử dụng chưa?

public bool dasudungkhckt(string s)

//Hàm xác định tập ký hiệu chưa kết thúc chưa sử dụng

public IList<string> tapkhcktchuasudung()

//Hàm kiểm tra văn phạm có sản xuất mà vế trái bằng về phải hay không?

public bool kosxvetraibangvephai()

//Hàm xác định tập ký hiệu đạt đến được

public IList<string> tapdatdenduoc()

//Hàm xác định tập ký hiệu không đạt đến được

public IList<string> tapkodatdenduoc()

//Hàm kiểm tra tính đúng của văn phạm phi ngữ cảnh

public bool kiemtratinhdung()

//Hàm tính First

public string timfirst(string s)

//Hàm tính Follow

public string timfollow(string s)

## Thuật toán

* + 1. **Kiểm tra văn phạm:**

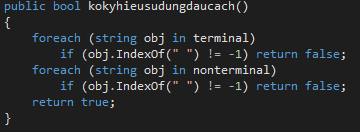
Trước khi đi vào tìm First và Follow, việc kiểm tra tính đúng đắn của văn phạm là cần thiết và cần được thực hiện đầu tiên. Ngôn ngữ có thể được biểu diễn bởi nhiều cách khác nhau, tuy nhiên đối với các ngôn ngữ có cấu trúc thì cần phải có một văn phạm đúng để biểu diễn. Một văn phạm phi ngữ cảnh đúng mới có thể sản sinh được ngôn ngữ. Có nhiều phương pháp nhằm kiểm tra tính đúng của văn phạm phi ngữ cảnh và một phương pháp trong số đó là từng bước kiểm tra tính chính xác của các thành phần tạo nên văn phạm. Phương pháp đề xuất bao gồm các công việc chính sau: kiểm tra cấu tạo của các ký hiệu kết thúc và ký hiệu chưa kết thúc, kiểm tra việc sử dụng các ký hiệu này để tạo ra các sản xuất, kiểm tra các sản xuất có vi phạm các điều kiện để tạo ra được ngôn ngữ hay không? Giải pháp đề xuất có thể ứng dụng để kiểm tra một văn phạm phi ngữ cảnh trước khi hoạt động để sản sinh ra ngôn ngữ, phục vụ cho việc tính toán sau đó.

Một chương trình là một dãy các câu lệnh, một câu lệnh là một dãy các từ, một từ được tạo ra từ các ký tự. Chương trình, câu lệnh, từ (hay còn gọi là xâu) được tạo ra theo một qui tắc nhất định, qui tắc đó chính là văn phạm. Hay nói cách khác, văn phạm là cơ chế để sản sinh ra ngôn ngữ. Chương trình trước khi được thực thi thì phải thông qua giai đoạn biên dịch để kiểm tra xem nó có bị các lỗi từ vựng hay lỗi cú pháp không. Để việc kiểm tra được chính xác, đòi hỏi việc xây dựng văn phạm để tạo ra ngôn ngữ lập trình cũng phải chính xác.

Như đã nói ở phần khái niệm thì một văn phạm phi ngữ cảnh đúng thì mới sản sinh ra được ngôn ngữ. Văn phạm đó phải thỏa mãn những điều kiện sau:

* **Không sử dụng dấu cách trong ký hiệu**

Ta có hàm kiểm tra tất cả các ký hiệu của văn phạm có sử dụng dấu cách không như sau:

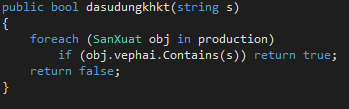


Hàm này sẽ cho xét hết từng chuỗi ký hiệu có ở trong tập các ký hiệu kết thúc, sử dụng hàm IndexOf để tìm trong từng chuỗi ký hiệu lấy ra đó có sự hiện diện của dấu cách không. Nếu như có thì kết quả sẽ trả về một số nguyên dương là vị trí của dấu cách trong chuỗi và do đó hàm sẽ lập tức kết thúc và trả về giá trị False.

Nếu như tập các ký hiệu kết thúc không có sử dụng dấu cách thì tiếp tục làm tương tự xét tiếp đến tập các ký hiệu chưa kết thúc. Nếu vẫn không có ký hiệu nào có sử dụng dấu cách thì kết thúc hàm và trả về giá trị True.

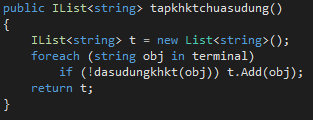
* **Không ký hiệu kết thúc hoặc chưa kết thúc chưa được sử dụng trong sản xuất**

Ta có hàm kiểm tra ký hiệu kết thúc s bất kỳ đã có hay chưa như sau:



Hàm này sẽ cho xét hết tất cả các sản xuất và tìm trong số đó xem thử có chứa ký hiệu s không nếu như có thì ngay lập tức hàm trả về giá trị True còn nếu như đã xét hết tất cả sản xuất mà vẫn không tồn tại ký hiệu s thì hàm sẽ trả về giá trị False.

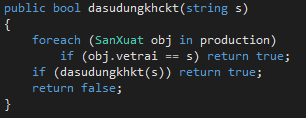
Dựa vào hàm trên ta có thể áp dụng để tìm ra tập các ký hiệu kết thúc A mà chưa được sử dụng bằng hàm tổng quát hơn như sau:



Trước tiên ta tạo mới một biến t là list string. Với mỗi chuỗi ký hiệu trong tập các ký hiệu kết thúc terminal thì ta sử dụng lại hàm kiểm tra đã nói ở trên để kiểm tra, nếu kết quả trả về là False tức là chuỗi ký hiệu đó chưa được sử dụng trong các sản xuất thì ta thêm chuỗi ký hiệu đó vào list t. Sau khi chạy xong vòng lặp thì giá trị trả về của hàm sẽ là một list chuỗi t chứa các ký hiệu kết thúc chưa được sử dụng.

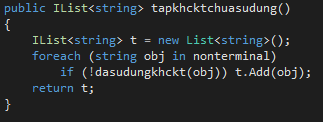
Sau khi đã kiểm tra xong với ký hiệu kết thúc, ta lặp lại quá trình trên với tín hiệu chưa kết thúc.

Tương tự như trên ta có hàm kiểm tra tín hiệu chưa kết thúc s bất kỳ đã có hay chưa:



Hàm này sẽ cho xét hết tất cả các sản xuất và tìm trong số đó xem thử có sản xuất nào có vế trái chính là s thì hàm lập tức trả về giá trị True, nếu như không có trường hợp nào thì hàm sẽ gọi lại hàm dasudungkhkt() đã khai báo vừa nãy để kiểm tra tương tự như đối với ký hiệu kết thúc mà ta đã đề cập ở trên.

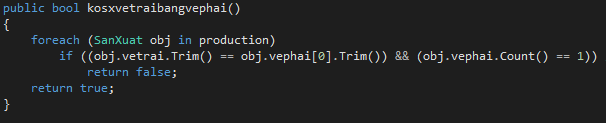
Tương tự như trường hợp ký hiệu kết thúc ta cũng có một hàm để tìm ra tập các ký hiệu chưa kết thúc A mà chưa được sử dụng bằng hàm tổng quát hơn như sau:



Thay vì tìm trong tập các ký hiệu kết thúc như đối với trường hợp ký hiệu kết thúc ở trên thì ở trường hợp này ta cũng làm tương tự như trên nhưng thay vào đó tìm trong tập các ký hiệu chưa kết thúc nonterminal và do đó kết quả trả về của hàm sẽ là một list chuỗi t chứa các ký hiệu chưa kết thúc mà chưa được sử dụng.

* **Không có sản xuất vế phải bằng vế trái**

Ta có hàm kiểm tra văn phạm có sản xuất mà vế trái bằng vế phải hay không như sau:

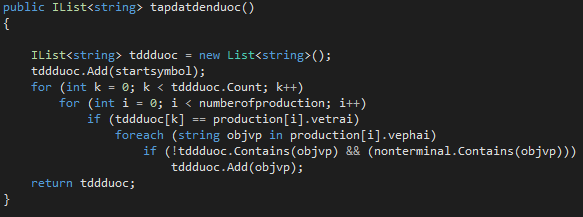


Hàm này sẽ xét từng sản xuất trong mảng các sản xuất, với mỗi sản xuất ta lấy vế trái ra để so sánh và nếu sản xuất nào có vế trái bằng với phần tử đầu tiên của vế phải đồng thời số phần tử của vế phải của sản xuất đó bằng 1 thì hàm sẽ lập tức trả về giá trị False, còn nếu như không có sản xuất nào thỏa điều kiện trên thì hàm sẽ trả về giá trị True.

* **Không có ký hiệu không đạt đến được**

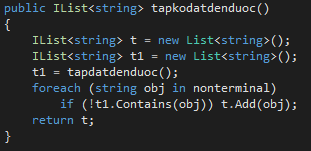
Một ký hiệu A∈ Δ được gọi là ký hiệu không đạt đến được khi S ≠˃+αAβ

Để tìm ra được tập các ký hiệu không đạt được thì trước hết ta sử dụng một hàm xác định tập các ký hiệu đạt đến được như sau:



Trước tiên ta tạo mới một biến tddduoc là một list string chính là tập các ký hiệu đạt đến được. Và vì ký hiệu bắt đầu luôn là một ký hiệu đạt đến được nên ta thêm nó vào tddduoc trước tiên. Sau đó ta cho chạy một vòng for chạy duyệt qua tất cả các phần tử của tddduoc, với mỗi phần từ của tddduoc ta cho một vòng for khác duyệt qua tất cả các sản xuất để xem thử liệu có vế trái của sản xuất nào bằng với phần tử của tddduoc không. Nếu có thì ta xét lần lượt từng chuỗi ký hiệu trong vế phải của sản xuất đó để xem thử là phần tử của tddduoc có chứa chuỗi ký hiệu đó không, nếu như không chứa và đồng thời chuỗi ký hiệu đó là ký hiệu không kết thúc thì nó được xem là một ký hiệu đạt đến được và do đó ta thêm chuỗi ký hiệu đó vào tddduoc. Giá trị trả về của hàm sẽ là một list các ký hiệu mà có thể đạt đến được.

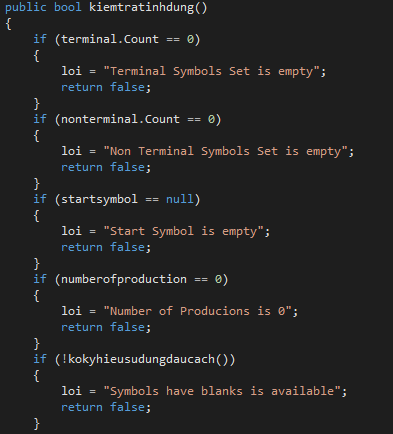
Sau khi đã có được tập các ký hiệu đạt đến được ta dùng hàm sau để loại trừ các ký hiệu đó ra và từ đó tìm ra được tập các ký hiệu không đạt đến được:



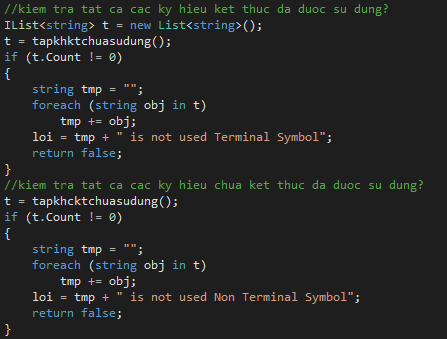
Hàm trên khởi tạo hai list string t và t1 với t1 là tập các ký hiệu đạt đến được và t là tập các ký hiêu không đạt đến được. Các giá trị của t sẽ được thêm vào nào như nó không thuộc vào tập các ký hiệu đạt đến được tức là không thuộc t1.

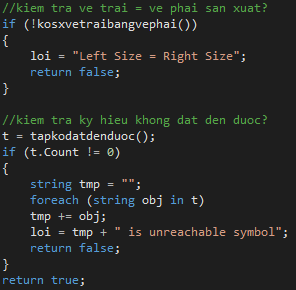
**Tổng hợp:**

Sau khi đã cài đặt cho tất cả các điều kiện mà văn phạm cần thỏa mãn. Việc cuối cùng cần làm là kết hợp tất cả các hàm lại với nhau để qua đó kiểm tra tính đúng đắn của một văn phạm phi ngữ cảnh. Hàm tổng hợp sẽ như sau:



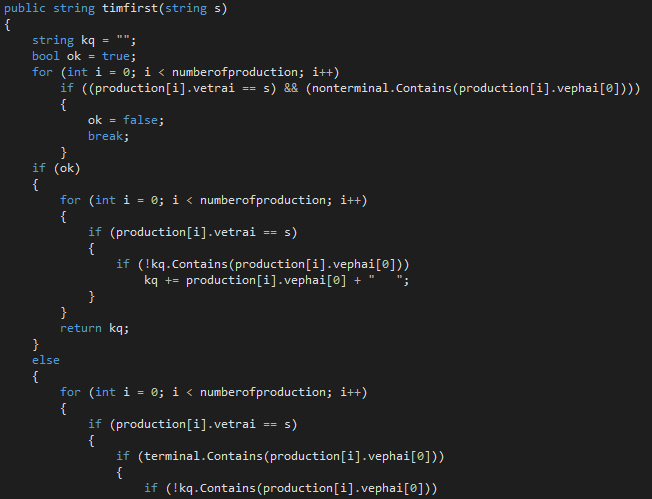
Các phép so sánh đầu tiên của hàm sẽ dùng để tìm lỗi trong các biến cơ bản của văn phạm. Để việc kiểm tra tính đúng đắn của văn phạm có thể được thực hiện thì trước nhất người sử dụng sẽ được yêu cầu nhập vào các tham số cơ bản là tập các ký hiệu kết thúc, tập các ký hiệu chưa kết thúc, ký hiệu bắt đầu. Nếu như số ký hiệu kết thúc hay số ký hiệu chưa kết thúc mà bằng 0 thì lỗi được trả về sẽ là thông báo cho người dùng biết tập ký hiệu tương ứng đang rỗng. Tương tự đối với trường hợp của ký hiệu bắt đầu hay số lượng sản xuất của văn phạm trong trường hợp nó rỗng hoặc bằng 0.

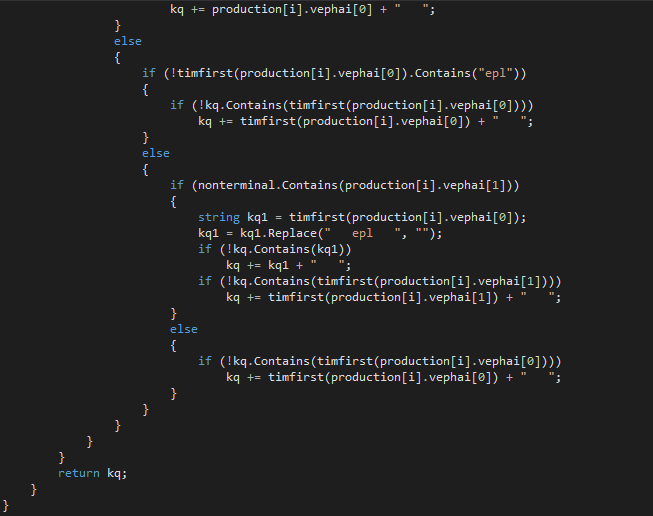




Các phép so sánh tiếp theo dùng để kiểm tra những điều kiện cần có để thỏa mãn tính đúng đắn của văn phạm mà đã được liệt kê ở trên. Lần lượt là kiểm tra các tín hiệu kết thúc và tín hiệu chưa kết thúc đã được sử dụng chưa, kiểm tra xem có sản xuất nào có vế trái bằng vế phải không và kiểm tra xem có ký hiệu nào không đạt đến được không. Các lỗi nếu có sẽ được lần lượt kiểm tra và nếu có thì sẽ được lưu vào biến loi, biến loi sẽ bao gồm ký hiệu cụ thể kèm theo lỗi của nó nếu như trường hợp kiểm tra lỗi tương ứng có giá trị trả về là một hoặc nhiều chuỗi ký hiệu.

### Tìm First:

Ta có thuật toán tìm First như sau:



Khởi đầu hàm ta tạo sẵn một biến kq là biến string dùng để lưu kết quả là First của ký hiệu s đầu vào và một biến kiểu bool là ok với giá trị mặc định ban đầu là True. Đầu tiên ta sẽ cho chạy vòng lặp for duyệt qua tất cả các sản xuất, với mỗi sản xuất ta xét xem vế trái của nó có bằng s không và đồng thời vế phải của nó có phải là ký hiệu chưa chưa kết thúc không, nếu có thì ta gán giá trị của biến ok là False và ngay lập tức thoát khỏi vòng lặp. Mục đích của vòng lặp này là để tìm ra liệu First của s lúc này có phải toàn là các ký hiệu kết thúc được suy dẫn ra trực tiếp từ s không, nếu phải thì giá trị của biến ok sẽ là True còn nếu không phải tức là First của s sẽ bằng First của một ký hiệu chưa kết thúc khác được suy dẫn từ s thì giá trị của biến ok sẽ là False.

Sau đó ta sẽ tìm First của ký hiệu s dựa vào giá trị biến ok vừa tìm được. Nếu giá trị của biến ok là True tức là s suy dẫn ngay các giá trị kết thúc thì ta cho vòng lặp duyệt hết các sản xuất, nếu có sản xuất nào có vế trái là s thì xét xem coi thử phần tử đầu tiên của vế phải đã có trong biến kq chưa, nếu chưa có thì ta thêm nó vào biến kq.

Nếu giá trị của biến ok là false tức là First của s sẽ là First của một ký hiệu chưa kết thúc khác được suy dẫn từ s thì trước tiên ta cho thực hiện một vòng lặp for duyệt qua tất cả các sản xuất để tìm ra sản xuất nào có vế trái là s. Sau đó với mỗi sản xuất tìm được ta kiểm tra phần tử đầu tiên của vế phải, nếu đó là một ký hiệu kết thúc và nó chưa có trong biến kq thì ta thêm ký hiệu đó vào biến kq. Còn nếu đó là một ký hiệu chưa kết thúc thì ta sẽ thực hiện đệ quy gọi lại hàm timfirst cho ký hiệu chưa kết thúc này và dựa vào kết quả tìm được ta sẽ phân làm hai trường hợp như sau:

**Trường hợp 1:** First của ký hiệu tìm được không có chứa epsilon (ε)

Trong trường hợp này ta chỉ việc kiểm tra bình thường xem kết quả tìm được đã có trong biến kq chưa và nếu chưa thì ta thêm nó vào biến kq.

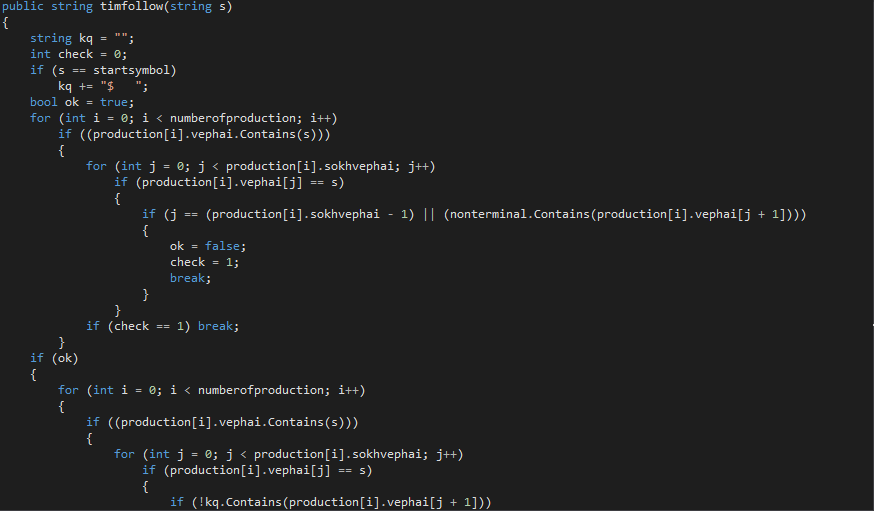
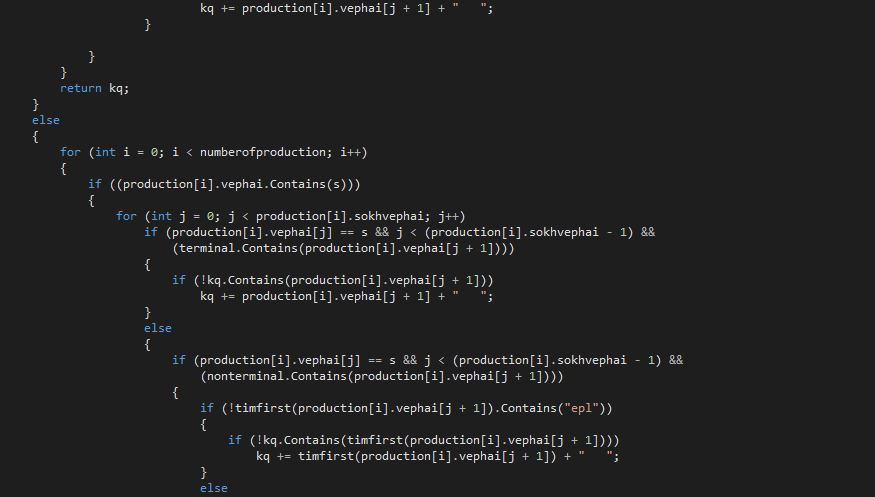
**Trường hợp 2:** First của ký hiệu tìm được có chứa epsilon (ε)

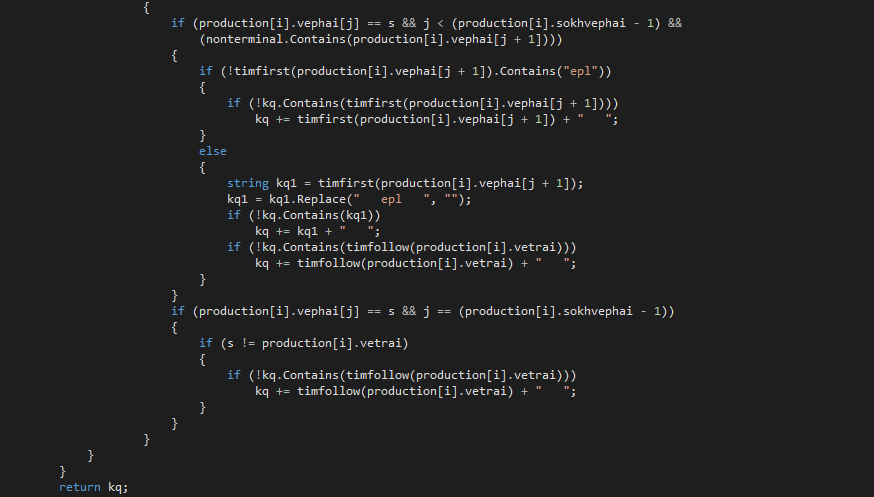
Nếu rơi vào trường hợp này thì trước tiên ta kiểm tra xem thử ký hiệu kế tiếp của ký hiệu mà ta gọi lại timfirst ở trên là ký hiệu gì. Nếu như cũng là ký hiệu chưa kết thúc thì trước tiên ta lưu các giá trị tìm được của hàm timfirst của ký hiệu đầu tiên mà có chứa ε vào một biến mới kiểu string là kq1, sau đó ta loại bỏ ký hiệu ε khỏi biến này và kiểm tra giá trị sau khi đã loại bỏ ε xem thử nó đã có trong biến kq chưa, nếu chưa thì ta thêm vào biến kq. Tiếp theo ta gọi đệ quy hàm timfirst cho ký hiệu chưa kết thúc tiếp theo và kiểm tra xem kết quả tìm được đã có trong biến kq để thêm vào biến kq.

Nếu như ký hiệu kế tiếp của ký hiệu mà ta gọi lại timfirst mà không phải là ký hiệu chưa kết thúc thì chỉ việc gọi đệ quy hàm timfirst cho ký hiệu này và kiểm tra xem kết quả tìm được đã trong biến kq chưa, nếu chưa thì thêm nó vào biến kq.

Hàm sẽ kết thúc với giá trị trả về là biến kq tìm được.

### Tìm Follow:

Ta có hàm tìm Follow như sau:



Tương tự như hàm tìm First, bắt đầu hàm tìm Follow ta cũng tạo mới hai biến là biến string kq dùng để lưu giá trị tìm được và biến bool ok đồng thời tạo thêm một biến kiểu int là check. Nếu như s là ký hiệu bắt đầu thì ta thêm ký hiệu đô la ($) vào biến kq. Trước tiên ta cho chạy hai vòng lặp for, vòng lặp for đầu tiên xét qua tất cả các sản xuất để tìm ra sản xuất nào có vế phải có tồn tại ký hiệu s ở trong không, nếu có thì ta sẽ cho chạy vòng lặp thứ hai xét lần lượt từng phần tử của vế phải có chứa s của sản xuất đó để tìm ra vị trí của ký hiệu s trong vế phải đó. Nếu vị trí của s trong vế phải mà chính là vị trí cuối cùng của vế phải hoặc nếu như không phải là vị trí cuối cùng nhưng ký hiệu ở vị trí tiếp theo lại là một ký hiệu chưa kết thúc thì ngay lập tức ta thoát ra vòng lập và gán trị cho biến ok là False và gán giá trị cho biến check là 1. Biến check là 1 cho vòng lặp đầu tiên biết được là vòng lặp thứ hai có thỏa điều kiện và có thay đổi biến ok do đó vòng lặp đầu tiền cũng sẽ kết thúc theo. Mục đích của hai vòng lặp này là để kiểm tra xem liệu Follow của s có thể tìm được là các ký hiệu kết thúc mà không cần thông qua một Follow của ký hiệu chưa kết thúc khác được suy dẫn ra từ s (giá trị biến ok là True) hay không.

Nếu giá trị biến ok là True tức là Follow của s có thể tìm ra được ngay là các ký hiệu kết thúc mà không cần thông qua Follow của các ký hiệu khác thì ta cho chạy hai vòng lặp for như lúc nãy, vòng lặp for đầu để tìm ra sản xuất nào có vế phải chứa s và vòng lặp for để tìm ra vị trí của s trong vế phải của sản xuất đó, khi tìm được vị trí của s rồi ta kiểm tra xem thử ký hiệu kế tiếp s trong vế phải của sản xuất đó đã có trong biến kq chưa, nếu chưa thì thêm ký hiệu đó vào biến kq.

Trong trường hợp giá trị biến ok là False tức là để tìm được Follow của s ta phải tìm Follow của ký hiệu chưa kết thúc được suy dẫn ra từ s. Trước tiên ta chạy lại hai vòng lặp for đã dùng ở trên để tìm sản xuất có vế phải có chứa s và vị trí của s trong vế phải đó. Tại đây ta kiểm tra nếu như vị trí của s không phải là vị trí của ký hiệu cuối cùng của vế phải và đồng thời ký hiệu ở ngay sau s chính là ký hiệu kết thúc thì ta kiểm tra ký hiệu ký hiệu kết thúc đó đã có ở trong biến kq chưa, nếu chưa thì ta thêm ký hiệu kết thúc đó vào biến kq.

Trong trường hợp không phải, tức là s không phải là ký hiệu cuối cùng của vế phải của sản xuất nhưng ký hiệu ngay sau s là ký hiệu chưa kết thúc hoặc s chính là ký hiệu cuối cùng của vế phải của sản xuất thì ta sẽ lần lượt xét qua cả hai trường hợp. Nếu s là ký hiệu cuối cùng của vế phải thì ta kiểm tra coi lúc đó s có trùng với ký hiệu của vế trái không, nếu không trùng thì ta kiểm gọi đệ quy hàm timfollow cho ký hiệu ở vế trái của sản xuất và kiểm tra kết quả tìm được có ở trong biến kq chưa, nếu chưa thì ta thêm nó vào biến kq.

Nếu s không phải là ký hiệu cuối cùng của vế phải nhưng ký hiệu ở liền sau s lại là ký hiệu chưa kết thúc thì ta tìm gọi hàm timfirst cho ký hiệu này rồi xét hai trường hợp tương tự ở đối với hàm tìm first ở trên đó là:

**Trường hợp 1:** First của ký hiệu tìm được không có chứa epsilon (ε)

Trong trường hợp này ta chỉ việc kiểm tra bình thường xem kết quả tìm được đã có trong biến kq chưa và nếu chưa thì ta thêm nó vào biến kq.

**Trường hợp 2:** First của ký hiệu tìm được có chứa epsilon (ε)

Nếu rơi vào trường hợp này thì trước tiên ta lưu các giá trị tìm được của hàm timfirst của mà có chứa ε vào một biến mới kiểu string là kq1, sau đó ta loại bỏ ký hiệu ε khỏi biến này và kiểm tra giá trị sau khi đã loại bỏ ε xem thử nó đã có trong biến kq chưa, nếu chưa thì ta thêm vào biến kq. Tiếp theo ta gọi đệ quy hàm timfollow cho ký hiệu ở vế trái của sản xuất và kiểm tra xem kết quả tìm được đã có trong biến kq để thêm vào biến kq.

Hàm sẽ kết thúc với giá trị trả về là biến kq tìm được.

# CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

* 1. **Tổ chức chương trình**

Chương trình được tổ chức bởi các thành phần chính sau đây:

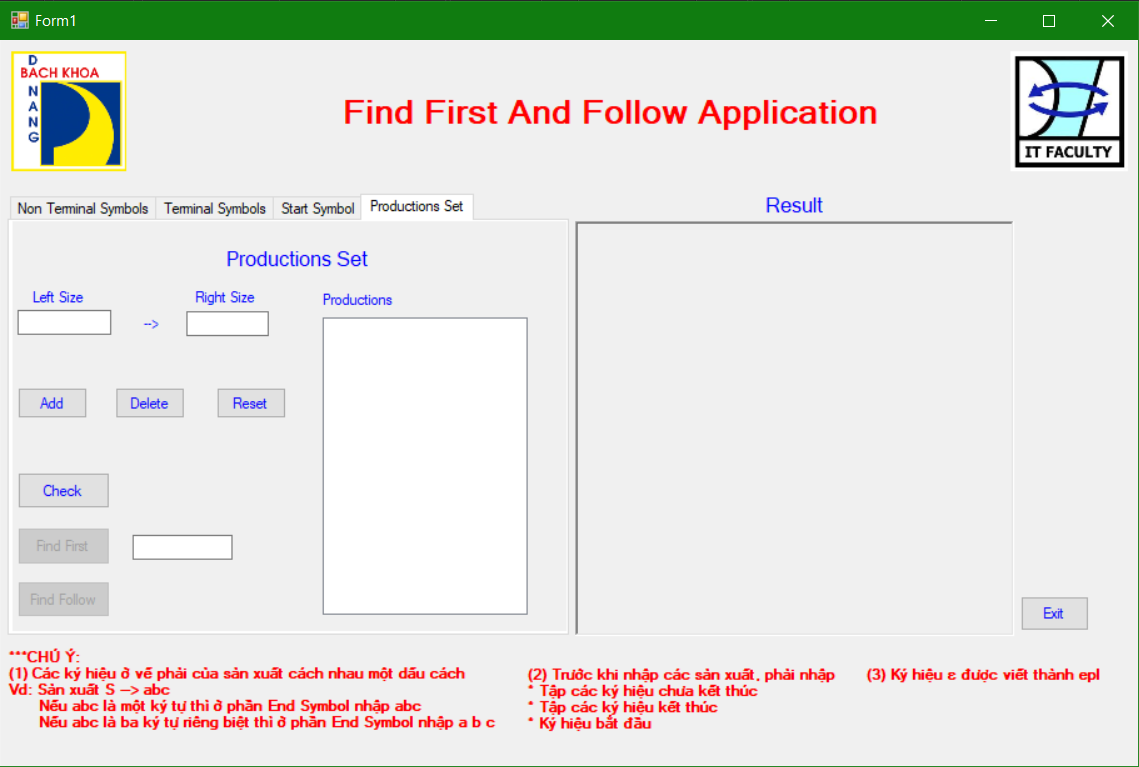
* **Struct SanXuat:** để lưu các sản xuất.
* **Class VanPham:** để lưu các văn phạm.
* **Hàm** **kiemtratinhdung:** để kiểm tra xem văn phạm phi ngữ cảnh G có đúng hay không.
* **Hàm timfirst:** để tìm các tập First của văn phạm.
* **Hàm timfollow:** để tìm các tập Follow của văn phạm.

Chi tiết các thành phần này xem thêm ở phần phụ lục

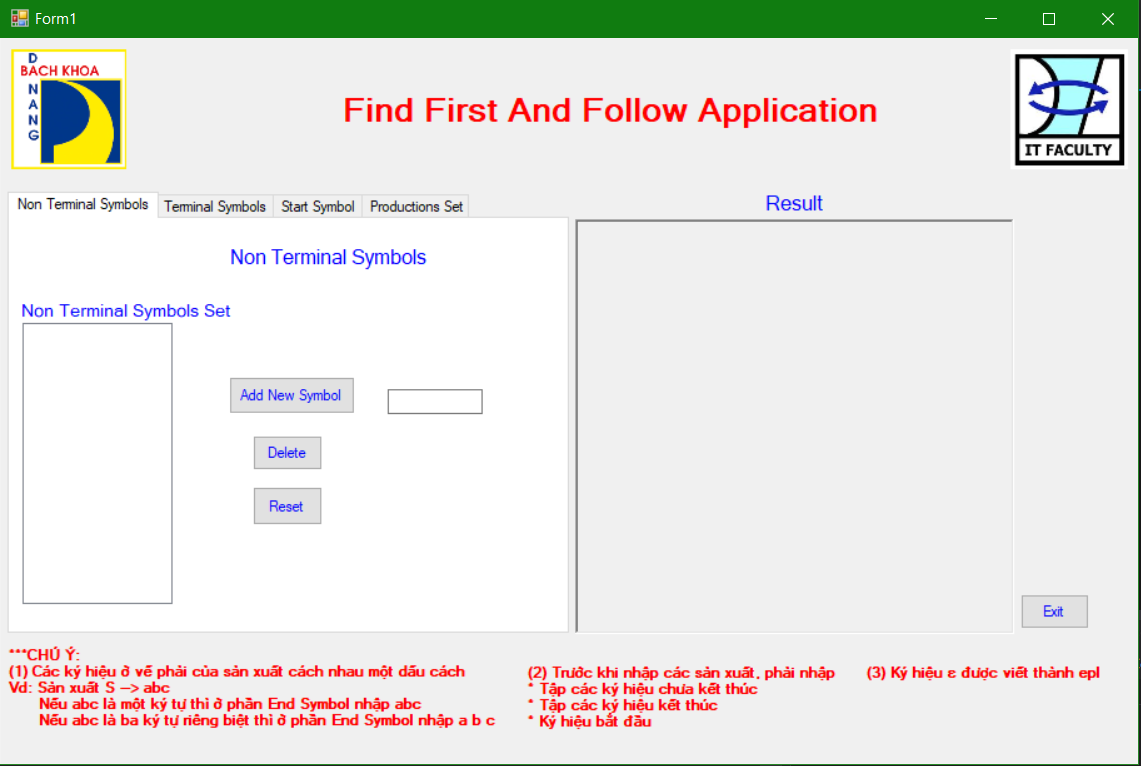
* 1. **Ngôn ngữ cài đặt**

Chương trình được cài đặt bằng ngôn ngữ lập trình C#, một ngôn ngữ khá quen thuộc và cơ bản đối với phần lớn sinh viên ngành kỹ thuật nói chung và sinh viên Công nghệ thông tin nói riêng. Sử dụng phần mềm Microsoft Visual Studio 2017 để viết chương trình và xây dựng ứng dụng Winform Application cho giao diện trực quan và dễ tương tác hơn.

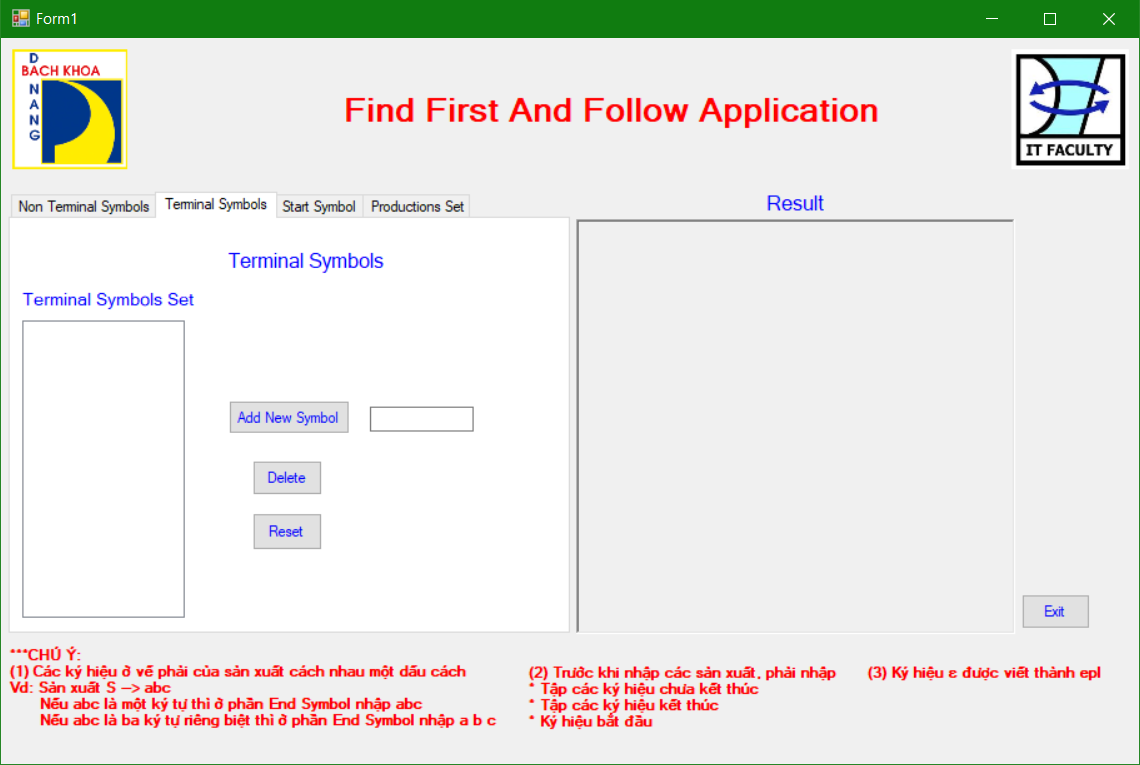
* 1. **Kết quả**
     1. **Giao diện chính của ứng dụng**

****

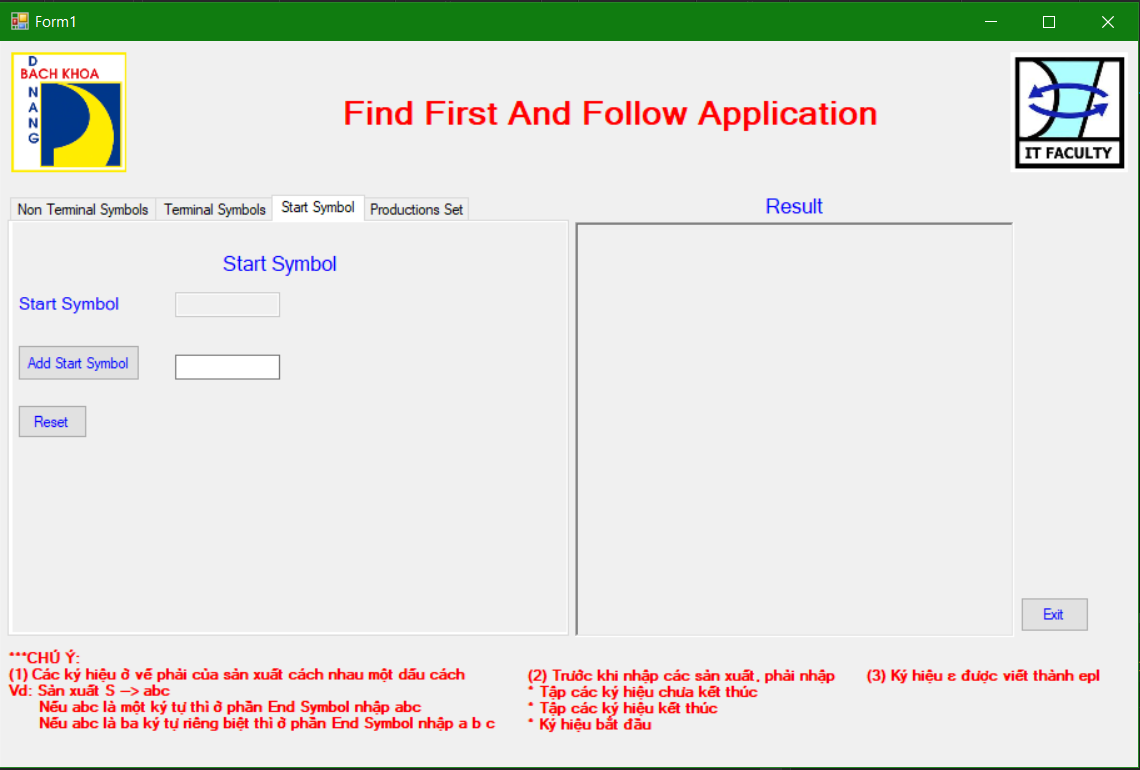
*Hình 2. Giao diện chính của ứng dụng*



*Hình 3. Giao diện chính của ứng dụng*



*Hình 4. Giao diện chính của ứng dụng*



*Hình 5. Giao diện chính của ứng dụng*

* + 1. **Kết quả thực thi của ứng dụng**

Chạy ứng dụng với ví dụ tính First và Follow được sử dụng ở phần 1.3.1.2 và 1.3.2.2

Văn phạm G

E → T E’

E’ → + T E’ |

T → F T’

T’ → \* F T’ |

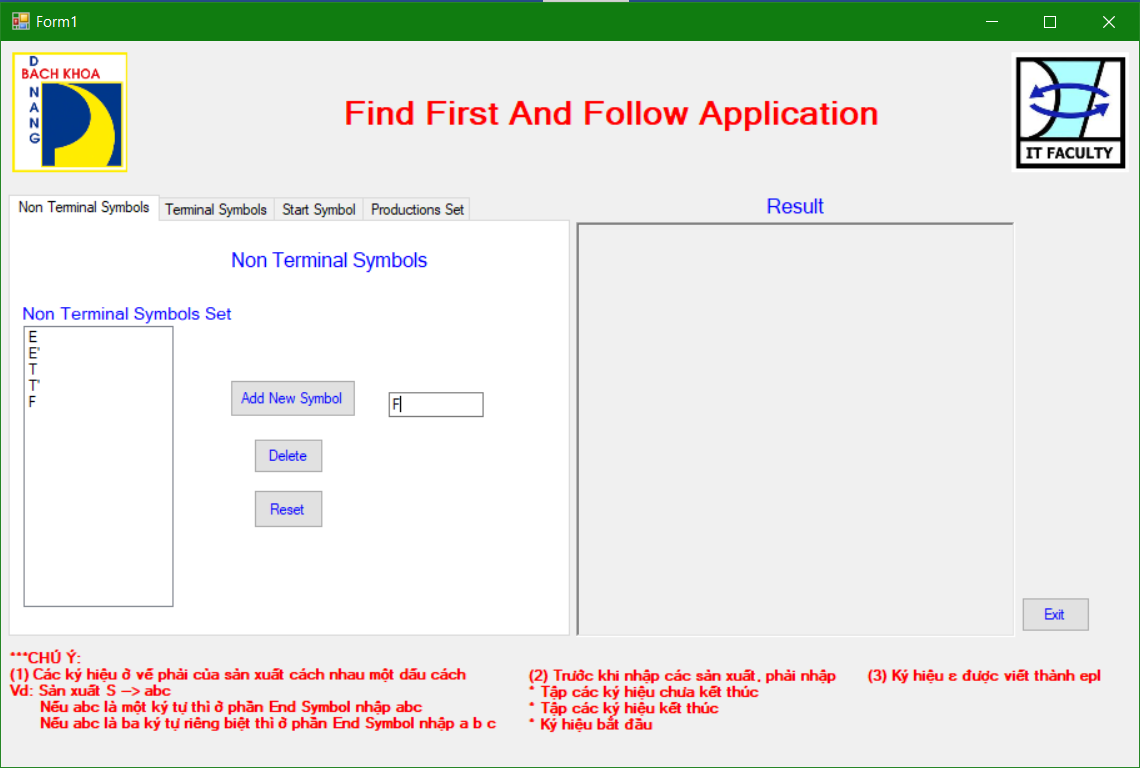
F → (E) | id

* + - 1. **Tính First của văn phạm G**

Văn phạm G trên có các thành phần:

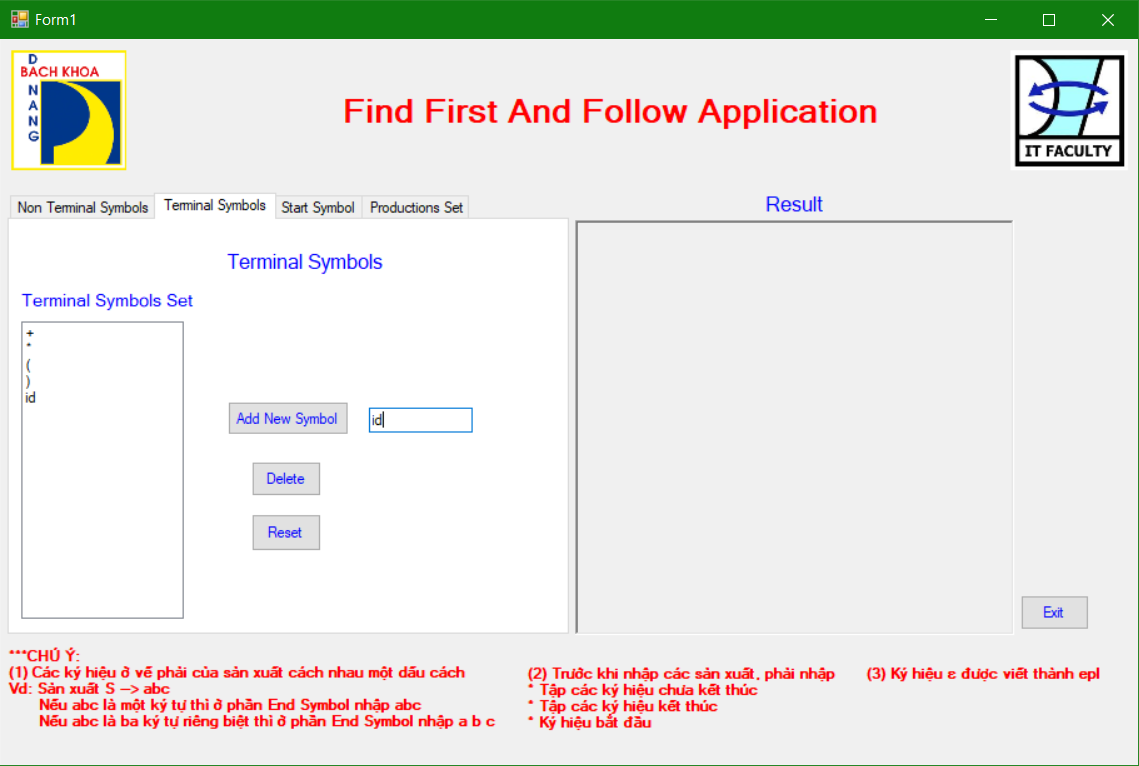
* Tập ký hiệu kết thúc ∑ bao gồm các ký hiệu: +, \*, (, ), id
* Tập ký hiệu chưa kết thúc ∆ bao gồm các ký hiệu: E, E’, T’, T’, F
* Ký hiệu bắt đầu S: E
* Tập các sản xuất P như trên

1. Nhập các ký hiệu chưa kết thúc vào tab Non Terminal Symbols



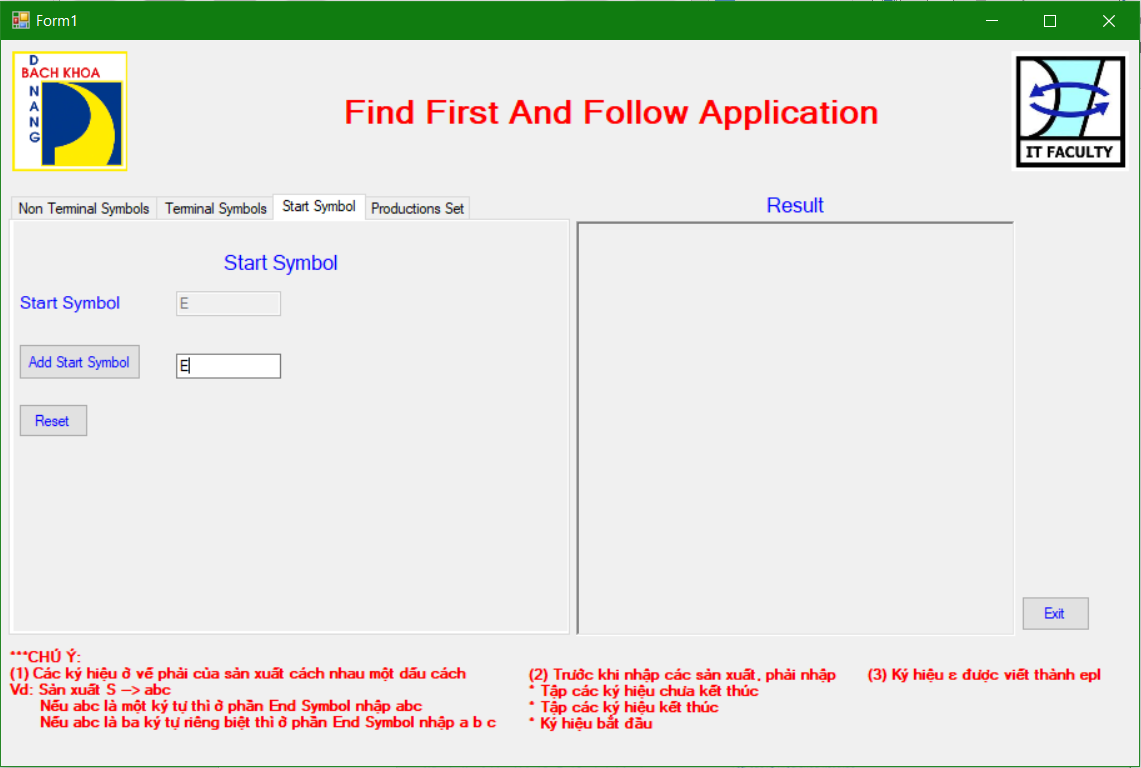
*Hình 6. Nhập các ký hiệu chưa kết thúc*

1. Nhập các ký hiệu chưa kết thúc vào tab Terminal Symbols



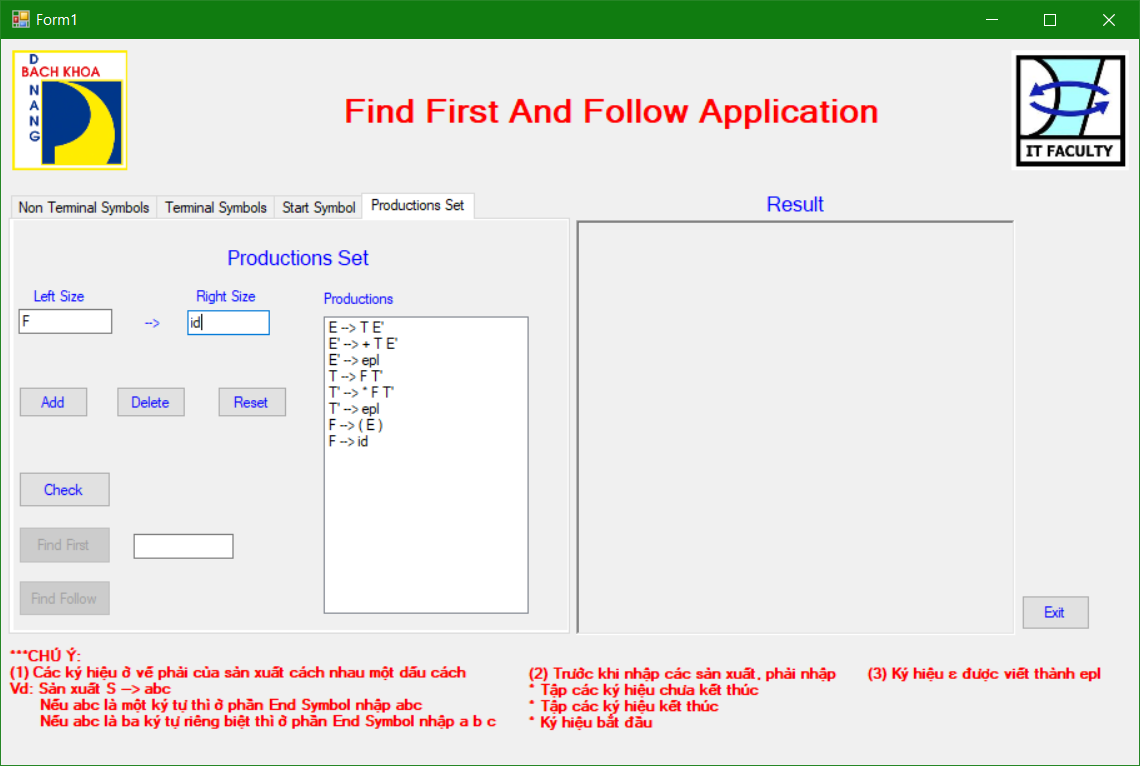
*Hình 7. Nhập các ký hiệu kết thúc*

1. Nhập ký hiệu bắt đầu vào tab Start Symbols



*Hình 8. Nhập ký hiệu bắt đầu*

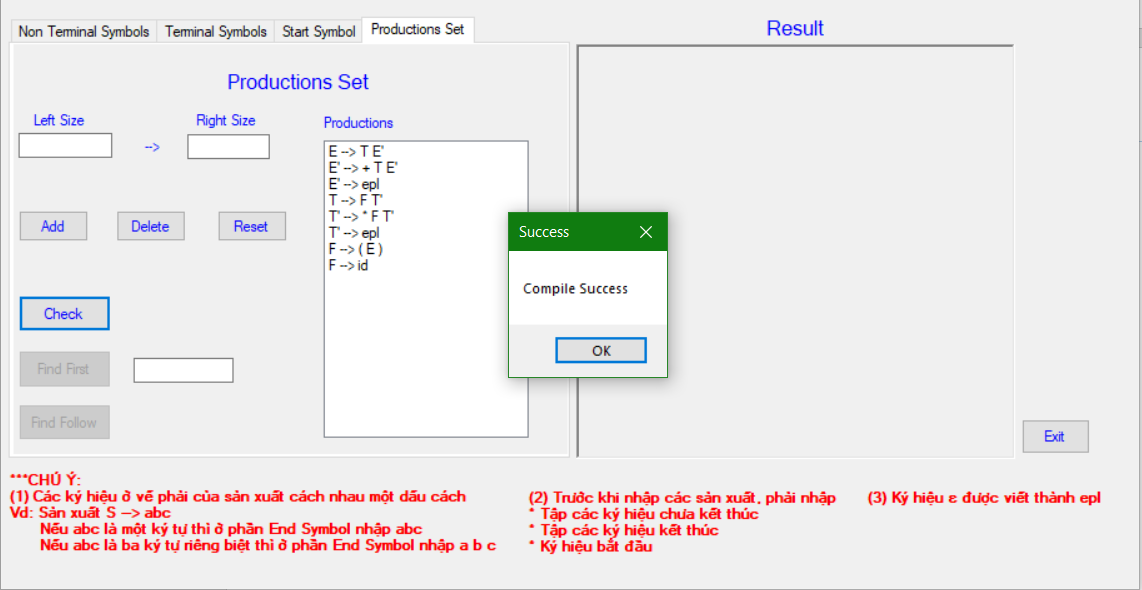
1. Nhập các sản xuất vào tab Productions Set



*Hình 9. Tập các sản xuất*

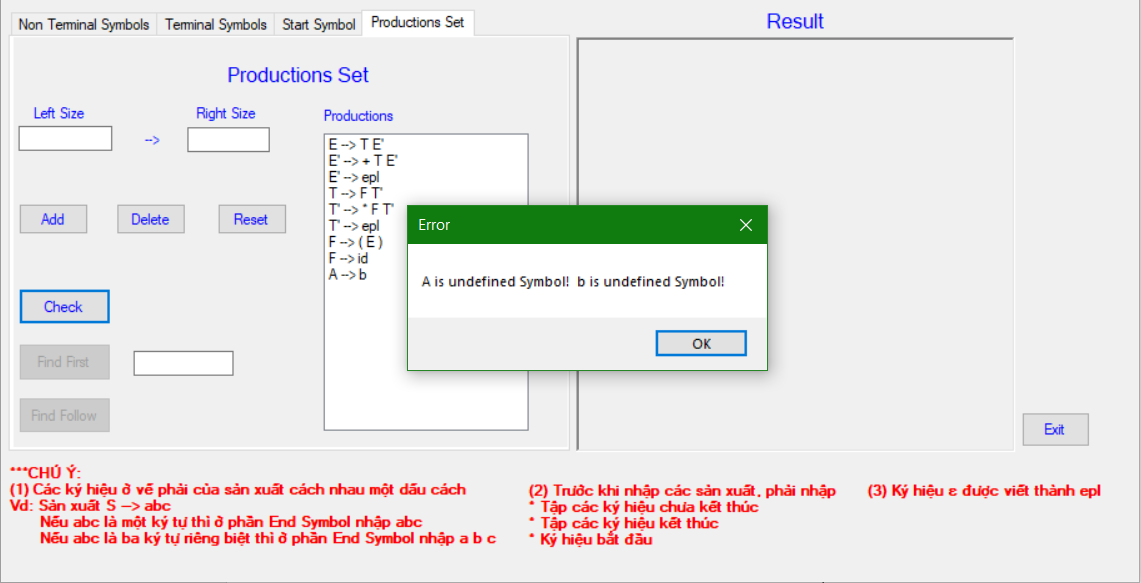
1. Bấm nút Check để kiểm tra tính đúng đắn của văn phạm, nếu văn phạm đúng thì thông báo Compile success và tiếp tục tìm First và Follow, nếu sai thì báo lỗi.

* Nếu kiểm tra văn phạm đúng



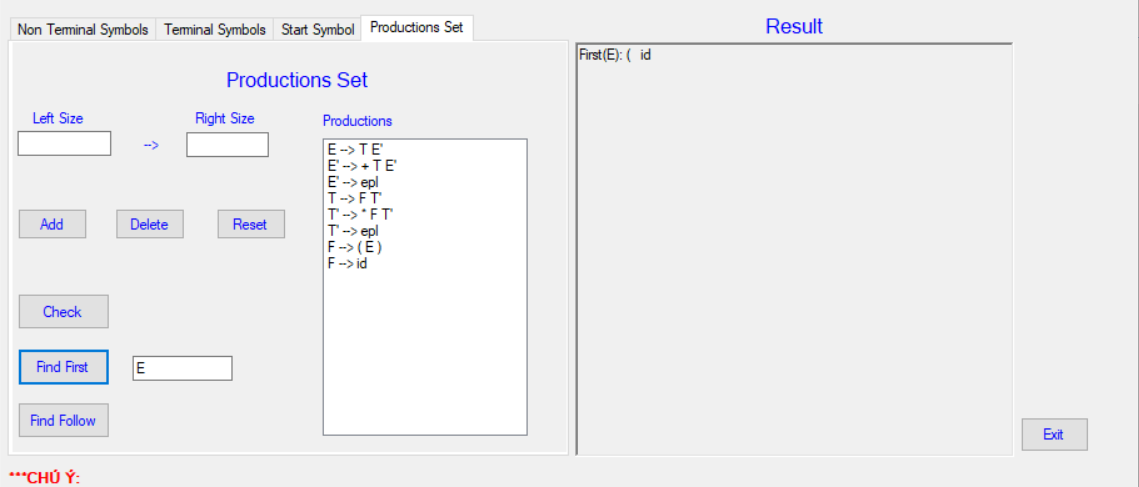
*Hình 10. Kiểm tra văn phạm đúng*

* Nếu kiểm tra văn phạm sai

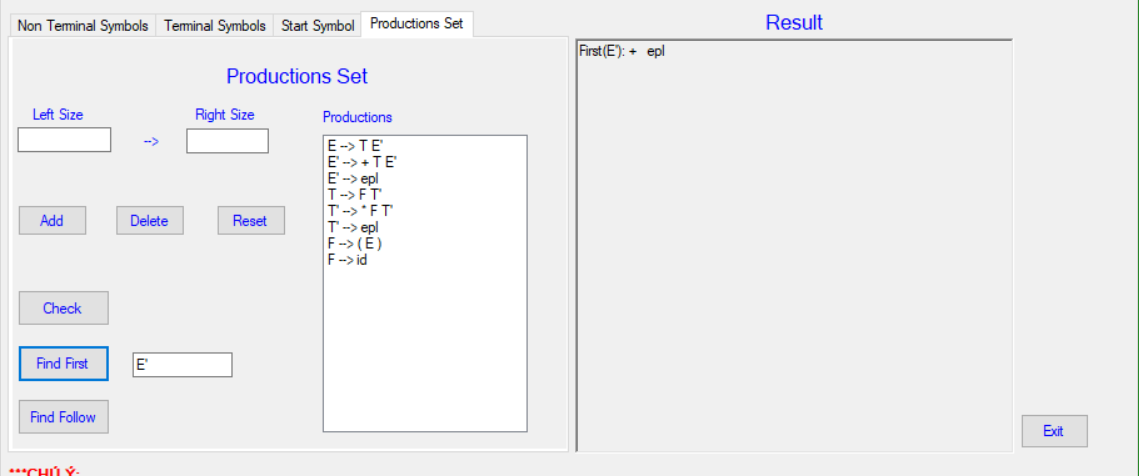


*Hình 11. Kiểm tra văn phạm sai*

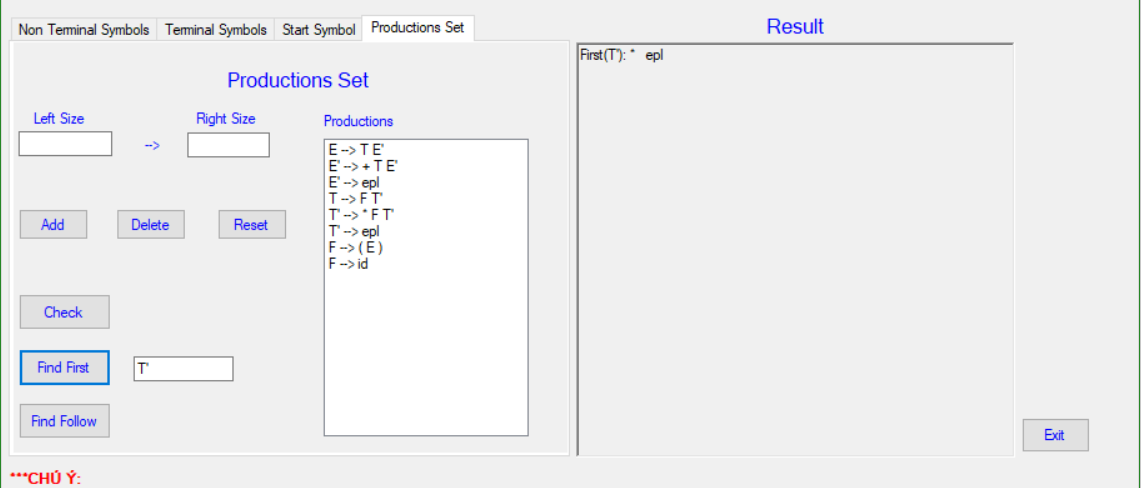
1. Tìm First, nhập giá trị cần tìm vào và nhấn nút Find First



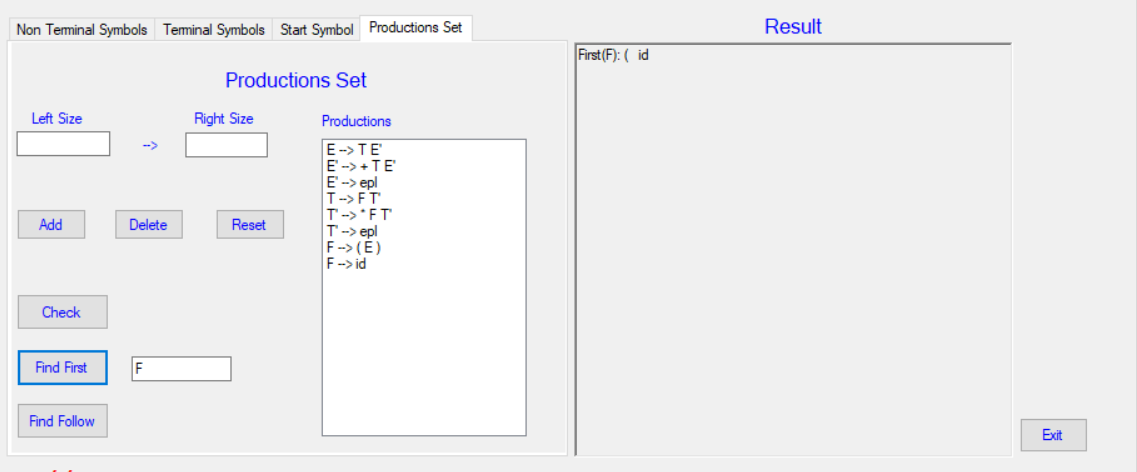
*Hình 12. First(E)*



*Hình 13. First(E’)*



*Hình 14. First(T’)*

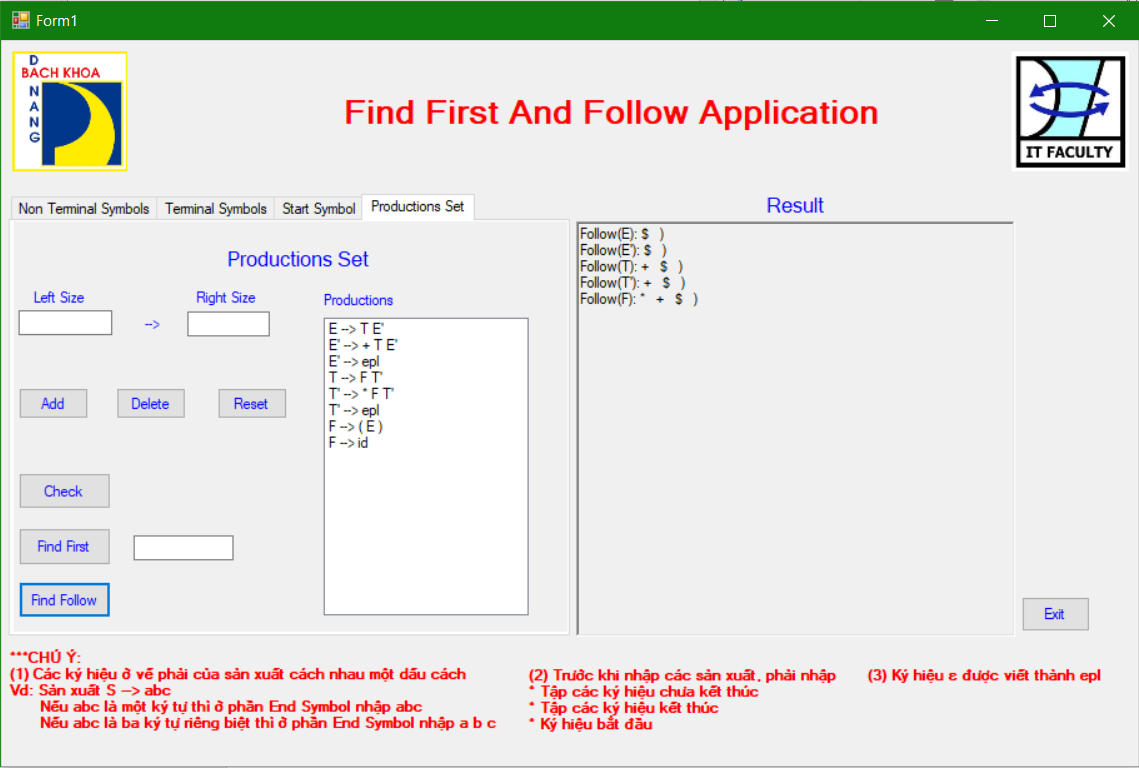


*Hình 15. First(F)*

* + - 1. **Tính Follow của văn phạm G**

Nhập và kiểm tra văn phạm G với các bước tương tự như tính First ở phần 3.3.2.1

Bấm vào nút Find Follow để tìm các tập Follow của văn phạm G



*Hình 16. Tập các Follow của văn phạm G*

* 1. **Nhận xét**

Ứng dụng đã giải quyết được phần lớn những yêu cầu cơ bản mà bài toán đặt ra. Các kết quả thực hiện đều thỏa mãn những tiêu chí ban đầu của bài toán, bên cạnh đó còn phát triển thêm một số tính năng bổ sung nhằm hoàn thiện yêu cầu bài toán.

Tốc độ thực thi bài toán nhanh do lượng dữ liệu không nhiều. Nếu số lượng các ký hiệu và các sản xuất tăng lên thì có thể ảnh hưởng đến tốc độ nhưng không đáng kể do giao diện đồ họa đơn giản và tối ưu các thuật toán, cấu trúc dữ liệu sử dụng trong chương trình.

# CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. **Kết luận**

Qua quá trình ngiên cứu và thực hiện đề tài đồ án, chúng em đã củng cố và có thêm nhiều kiến thức lý thuyết và thực tiễn về Văn phạm, về First và Follow cũng như các phương pháp tính First và Follow. Bên cạnh đó chúng em cũng được củng cố và ôn lại cũng như học tập thêm những kỹ thuật lập trình, cách tổ chức cấu trúc dữ liệu và giải thuật sao cho hiệu quả, góp phần hoàn thiện hơn những kỹ năng của bản thân.

1. **Hướng phát triển**

Thuật toán kiểm tra tính đúng đắn của văn phạm và thuật toán tính First và Follow được trình bày trong đồ án này tuy đã giải quyết được những yêu cầu đề ra nhưng hẳn vẫn còn rất nhiều thiếu xót và chưa đầy đủ. Ứng dụng trên có thể phát triển tiếp về mặt giao diện để có thể tổng quan dễ nhìn hơn, phát triển các chức năng nhập dữ liệu và đọc dữ liệu từ bên ngoài để giúp quá trình tạo văn phạm trở nên dễ dàng hơn, tiết kiệm được nhiều thời gian khi tính toán. Cần cải tiến về thuật toán để tối ưu khả năng xử lý khi sử dụng cho dữ liệu lớn hơn.

# TÀI LIỆU THAM KHẢO

1. Compilers Principles, Techniques and Tools, Alfred V. Aho – Ravi Sethi – Jeffrey D. Ullman.
2. Giải pháp kiểm tra tính đúng đắn của văn phạm phi ngữ cảnh, Nguyễn Thị Minh Hỷ - Trần Hồ Thủy Tiên, Tạp chí khoa học công nghệ Đại học Đà Nẵng số 7 (104) – 2016.
3. Tính First và Follow trong chương trình dịch

<https://cachhoc.net/2015/03/08/school-tinh-first-va-follow-trong-chuong-trinh-dich>

1. Chương trình dịch

<https://txnam.net/wp-content/uploads/txnam/Bai%20Giang/Chuong%20Trinh%20Dich%20K53II/Chuong%20trinh%20dich%20K53II%20-%2016.pdf>

# PHỤ LỤC

**SanXuat.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DoAn2\_WinForm\_Final\_v0

{

public struct SanXuat

{

public string vetrai;

public string[] vephai;

public int sokhvephai;

}

}

**VanPham.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DoAn2\_WinForm\_Final\_v0

{

public class VanPham

{

public IList<string> terminal;

public IList<string> nonterminal;

public string startsymbol;

public int numberofproduction;

public SanXuat[] production;

public string loi = "";

public bool kokyhieusudungdaucach()

{

foreach (string obj in terminal)

if (obj.IndexOf(" ") != -1) return false;

foreach (string obj in nonterminal)

if (obj.IndexOf(" ") != -1) return false;

return true;

}

public bool dasudungkhkt(string s)

{

foreach (SanXuat obj in production)

if (obj.vephai.Contains(s)) return true;

return false;

}

public IList<string> tapkhktchuasudung()

{

IList<string> t = new List<string>();

foreach (string obj in terminal)

if (!dasudungkhkt(obj)) t.Add(obj);

return t;

}

public bool dasudungkhckt(string s)

{

foreach (SanXuat obj in production)

if (obj.vetrai == s) return true;

if (dasudungkhkt(s)) return true;

return false;

}

public IList<string> tapkhcktchuasudung()

{

IList<string> t = new List<string>();

foreach (string obj in nonterminal)

if (!dasudungkhckt(obj)) t.Add(obj);

return t;

}

public bool kosxvetraibangvephai()

{

foreach (SanXuat obj in production)

if ((obj.vetrai.Trim() == obj.vephai[0].Trim()) && (obj.vephai.Count() == 1)) //Note

return false;

return true;

}

public IList<string> tapdatdenduoc()

{

IList<string> tddduoc = new List<string>();

tddduoc.Add(startsymbol);

for (int k = 0; k < tddduoc.Count; k++)

for (int i = 0; i < numberofproduction; i++)

if (tddduoc[k] == production[i].vetrai)

foreach (string objvp in production[i].vephai)

if (!tddduoc.Contains(objvp) && (nonterminal.Contains(objvp)))

tddduoc.Add(objvp);

return tddduoc;

}

public IList<string> tapkodatdenduoc()

{

IList<string> t = new List<string>();

IList<string> t1 = new List<string>();

t1 = tapdatdenduoc();

foreach (string obj in nonterminal)

if (!t1.Contains(obj)) t.Add(obj);

return t;

}

public bool kiemtratinhdung()

{

if (terminal.Count == 0)

{

loi = "Terminal Symbols Set is empty";

return false;

}

if (nonterminal.Count == 0)

{

loi = "Non Terminal Symbols Set is empty";

return false;

}

if (startsymbol == null)

{

loi = "Start Symbol is empty";

return false;

}

if (numberofproduction == 0)

{

loi = "Number of Producions is 0";

return false;

}

if (!kokyhieusudungdaucach())

{

loi = "Symbols have blanks is available";

return false;

}

//kiem tra tat ca cac ky hieu ket thuc da duoc su dung?

IList<string> t = new List<string>();

t = tapkhktchuasudung();

if (t.Count != 0)

{

string tmp = "";

foreach (string obj in t)

tmp += obj;

loi = tmp + " is not used Terminal Symbol";

return false;

}

//kiem tra tat ca cac ky hieu chua ket thuc da duoc su dung?

t = tapkhcktchuasudung();

if (t.Count != 0)

{

string tmp = "";

foreach (string obj in t)

tmp += obj;

loi = tmp + " is not used Non Terminal Symbol";

return false;

}

//kiem tra ve trai = ve phai san xuat?

if (!kosxvetraibangvephai())

{

loi = "Left Size = Right Size";

return false;

}

//kiem tra ky hieu khong dat den duoc?

t = tapkodatdenduoc();

if (t.Count != 0)

{

string tmp = "";

foreach (string obj in t)

tmp += obj;

loi = tmp + " is unreachable symbol";

return false;

}

return true;

}

public string timfirst(string s)

{

string kq = "";

bool ok = true;

for (int i = 0; i < numberofproduction; i++)

if ((production[i].vetrai == s) && (nonterminal.Contains(production[i].vephai[0])))

{

ok = false;

break;

}

if (ok)

{

for (int i = 0; i < numberofproduction; i++)

{

if (production[i].vetrai == s)

{

if (!kq.Contains(production[i].vephai[0]))

kq += production[i].vephai[0] + " ";

}

}

return kq;

}

else

{

for (int i = 0; i < numberofproduction; i++)

{

if (production[i].vetrai == s)

{

if (terminal.Contains(production[i].vephai[0]))

{

if (!kq.Contains(production[i].vephai[0]))

kq += production[i].vephai[0] + " ";

}

else

{

if (!timfirst(production[i].vephai[0]).Contains("epl"))

{

if (!kq.Contains(timfirst(production[i].vephai[0])))

kq += timfirst(production[i].vephai[0]) + " ";

}

else

{

if (nonterminal.Contains(production[i].vephai[1]))

{

string kq1 = timfirst(production[i].vephai[0]);

kq1 = kq1.Replace(" epl ", "");

if (!kq.Contains(kq1))

kq += kq1 + " ";

if (!kq.Contains(timfirst(production[i].vephai[1])))

kq += timfirst(production[i].vephai[1]) + " ";

}

else

{

if (!kq.Contains(timfirst(production[i].vephai[0])))

kq += timfirst(production[i].vephai[0]) + " ";

}

}

}

}

}

return kq;

}

}

public string timfollow(string s)

{

string kq = "";

int check = 0;

if (s == startsymbol)

kq += "$ ";

bool ok = true;

for (int i = 0; i < numberofproduction; i++)

if ((production[i].vephai.Contains(s)))

{

for (int j = 0; j < production[i].sokhvephai; j++)

if (production[i].vephai[j] == s)

{

if (j == (production[i].sokhvephai - 1) || (nonterminal.Contains(production[i].vephai[j + 1])))

{

ok = false;

check = 1;

break;

}

}

if (check == 1) break;

}

if (ok)

{

for (int i = 0; i < numberofproduction; i++)

{

if ((production[i].vephai.Contains(s)))

{

for (int j = 0; j < production[i].sokhvephai; j++)

if (production[i].vephai[j] == s)

{

if (!kq.Contains(production[i].vephai[j + 1]))

kq += production[i].vephai[j + 1] + " ";

}

}

}

return kq;

}

else

{

for (int i = 0; i < numberofproduction; i++)

{

if ((production[i].vephai.Contains(s)))

{

for (int j = 0; j < production[i].sokhvephai; j++)

if (production[i].vephai[j] == s && j < (production[i].sokhvephai - 1) && (terminal.Contains(production[i].vephai[j + 1])))

{

if (!kq.Contains(production[i].vephai[j + 1]))

kq += production[i].vephai[j + 1] + " ";

}

else

{

if (production[i].vephai[j] == s && j < (production[i].sokhvephai - 1) && (nonterminal.Contains(production[i].vephai[j + 1])))

{

if (!timfirst(production[i].vephai[j + 1]).Contains("epl"))

{

if (!kq.Contains(timfirst(production[i].vephai[j + 1])))

kq += timfirst(production[i].vephai[j + 1]) + " ";

}

else

{

string kq1 = timfirst(production[i].vephai[j + 1]);

kq1 = kq1.Replace(" epl ", "");

if (!kq.Contains(kq1))

kq += kq1 + " ";

if (!kq.Contains(timfollow(production[i].vetrai)))

kq += timfollow(production[i].vetrai) + " ";

}

}

if (production[i].vephai[j] == s && j == (production[i].sokhvephai - 1))

{

if (s != production[i].vetrai)

{

if (!kq.Contains(timfollow(production[i].vetrai)))

kq += timfollow(production[i].vetrai) + " ";

}

}

}

}

}

return kq;

}

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace DoAn2\_WinForm\_Final\_v0

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}