1. **GIỚI THIỆU PROLOG**

* Prolog là một ngôn ngữ lập trình được sử dụng nhiều trong các ứng dụng của trí tuệ nhân tạo và [ngôn ngữ học trong khoa học máy tính](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_h%E1%BB%8Dc_trong_khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh&action=edit&redlink=1" \o "Ngôn ngữ học trong khoa học máy tính (trang chưa được viết)) (đặc biệt là trong ngành [xử lý ngôn ngữ tự nhiên](https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%BD_ng%C3%B4n_ng%E1%BB%AF_t%E1%BB%B1_nhi%C3%AAn)). Mục tiêu của Prolog là giúp người dùng mô tả lại bài toán trên ngôn ngữ của logic, dựa trên đó, máy tính sẽ tiến hành suy diễn tự động dựa vào những cơ chế suy diễn có sẵn ([hợp nhất](https://vi.wikipedia.org/wiki/H%E1%BB%A3p_nh%E1%BA%A5t_(ph%C3%A9p_to%C3%A1n)" \o "Hợp nhất (phép toán)), [quay lui](https://vi.wikipedia.org/wiki/Quay_lui_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) và [tìm kiếm theo chiều sâu](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_s%C3%A2u" \o "Tìm kiếm theo chiều sâu)) để tìm câu trả lời cho người dùng. Prolog chủ yếu được thiết kế như một ngôn ngữ lập trình khai báo.
* Prolog là ngôn ngữ lập trình logic (Prolog = Programming in Logic) do GS. A. Colmerauer đưa ra lần đầu tiên vào năm 1972 tại trường Đại học Marseille, Pháp. Đến năm 1980, Prolog nhanh chóng được áp dụng rộng rãi. Cú pháp và ngữ nghĩa của Prolog đơn giản và sáng sủa, nó được người Nhật coi là một trong những nền tảng để xây dựng [máy tính thế hệ thứ năm](https://vi.wikipedia.org/w/index.php?title=M%C3%A1y_t%C3%ADnh_th%E1%BA%BF_h%E1%BB%87_th%E1%BB%A9_n%C4%83m&action=edit&redlink=1" \o "Máy tính thế hệ thứ năm (trang chưa được viết)) mà ở đó, thay vì phải mô tả cách giải quyết một bài toán trên máy tính, con người chỉ cần mô tả bài toán và máy tính sẽ hỗ trợ họ nốt phần còn lại. Prolog đã được cài đặt trên hầu hết các dòng máy tính Unix/Linux, Macintosh, Windows.

1. **ĐẶC ĐIỂM CHÍNH CỦA PROLOG**
2. **Nguyên lý lập trình logic trên Prolog:**

* Nguyên lý lập trình logic trên Prolog dựa trên các mệnh đề Horn (Horn logic).Một mệnh đề Horn biễu diễn một sự kiện hay một sự việc nào đó là đúng hoặc không đúng, xảy ra hoặc không xảy ra (có hoặc không có, ...).
* Ví dụ mệnh đề Horn:

+ Y là con mèo

+ Nếu A là cha của B, B là cha của C thì A là ông nội của C

+ X là người

+ Nếu trời mưa thì quần áo bị ướt

* Một hệ thống logic sẽ thực hiện chương trình theo cách «suy luận» - tìm kiếm  
  dựa trên vốn «hiểu biết» đã có là chương trình - cơ sở dữ liệu, để minh chứng câu  
  hỏi là một khẳng định, là đúng (Yes) hoặc sai (No).

1. **Cú pháp của Prolog:**

* Một mệnh đề Horn có dạng: *Head:-Body.* Trong đó, toán tử “-“ đại diện cho toán tử suy ra “🡪”, *Head* là một vị từ logic, còn *Body* có thể là rỗng hoặc là một tập các vị từ logic.
* Ví dụ:

+ chẵn(X) :- X chia\_dư 2 = 0 (X là số chẵn nếu X chia cho 2 dư 0)

+ lẻ(X) :- X chia\_dư 2 = 1 (X là số lẻ nếu X chia cho 2 dư 1)

+ đèn\_sáng(X) :- công\_tắc\_bật(X) (Nếu công tắc đèn bật thì đèn sáng)

+ quần\_áo\_ướt(X) :- trời\_mưa(X) (Nếu trời mưa thì quần áo ướt)

1. **Các thuật ngữ Prolog:**

* Một chương trình Prolog là một cơ sở dữ liệu gồm các *mệnh đề* (clause). Mỗi mệnh đề được xây dựng từ các *vị từ* (predicat). Một vị từ là một phát biểu nào đó về các đối tượng có giá trị chân *đúng* (true) hoặc *sai* (fail). Một vị từ có thể có các đối là các nguyên logic (logic atom).
* Mỗi nguyên tử (nói gọn) biểu diễn một quan hệ giữa các *hạng* (term). Như vậy, hạng và quan hệ giữa các hạng tạo thành mệnh đề.
* Hạng được xem là những đối tượng “dữ liệu” trong một trình Prolog. Hạng có thể là *hạng sơ cấp* (elementary term) gồm *hằng* (constant), *biến* (variable) và các *hạng phức hợp* (compound term).
* Các hạng phức hợp biểu diễn các đối tượng phức tạp của bài toán cần giải quyết thuộc lĩnh vực đang xét. Hạng phức hợp là một *hàm tử* (functor) có chứa các *đối* (argument), có dạng

***Tên\_hàm\_tử (Đối\_1, Đối\_2,…,Đối\_n)***

* Tên hàm tử là một chuỗi chữ cái và/hoặc chũ số được bắt đầu bởi một chữ cái thường. Các đối có thể là biến, hạng sơ cấp, hoặc hạng phức hợp. Trong Prolog, hàm tử đặc biệt “.” (dấu chấm) biểu diễn cấu trúc danh sách (list). Kiểu dữ liệu hàm tử tương tự kiểu bản ghi (record) và danh sách (list) tương tự kiểu mảng (array) trong các ngôn ngữ lập trình mệnh lệnh (C, Pascal...).
* Ví dụ:

+ f(5, a, b).

+ [a, b, c]

+ student(Tu, 2000, info, 2, address(227, ‘Nguyễn Văn Cừ’, ‘quận 5’)).

+ location(X,Y).

* Mệnh đề có thể là một sự kiện, một luật (hay quy tắc) hoặc một câu hỏi. Prolog quy ước viết sau mỗi mệnh đề một dấu chấm để kết thúc như sau:
* + Luật: <…> :- <…>.

+ Sự kiện: <…>. (tương ứng với luật <…> :- true. )

+ Câu hỏi: ?- <…>. (ở chế độ tương tác có dấu nhắc lệnh)

1. **Các kiểu dữ liệu trong Prolog:**

* Các kiểu dữ liệu trong Prolog được xây dựng từ các ký tự ASCII:

+ Các chữ cái in hoa A,B,…,Z và chữ cái in thường a,b,…,z

+ Các chữ số từ 0 đến 9

+ Các ký tự đặc biệt, chẳng hạn: + - \* / < > = : & ~

* Kiểu hằng số: Prolog sử dụng cả số nguyên và số thực. Cú pháp rất đơn giản. Tùy theo phiên bản cài đặt, Prolog có thể xử lý các miền số nguyên và miền số thực khác nhau.

+ Ví dụ:

* 1
* 3.14
* 0
* -12
* Kiểu hằng logic:Prolog dùng hai hằng logic có giá trị true và fail
* Kiểu hằng chuỗi kí tự: Các hằng là chuỗi (string) các ký tự được đặt giữa hai dấu nháy kép

+ Ví dụ:

* “cosoAI \#\@ data” chuỗi có tùy ý kí tự
* “” chuỗi rỗng (empty string)
* “ \” ” chuỗi chỉ có một dấu nháy kép
* Kiểu hằng nguyên tử: Là chuỗi kí tự ở một trong ba dạng:

+ Chuỗi gồm chữ cái, chữ số và ký tự \_ luôn luôn được bắt đầu bằng một chữ cái in thường. Ví dụ:

* maingoctu
* ngophuhuudaison2000
* nguyen\_bao\_long

+ Chuỗi các ký tự đặc biệt:

* …
* .:.
* ::= =
* =====>

+ chuỗi đặt giữa hai dấu nháy đơn (quote) được bắt đầu bằng chữ in hoa, dùng phân biệt với các tên biến:

* ‘Tu’
* ‘Dai Son’
* ‘BAOBAO’

1. **Biến trong Prolog:**

* Tên biến là một chuỗi ký tự gồm chữ cái, chữ số, bắt đầu bởi chữ hoa hoặc  
  dấu gạch dưới dòng. Ví dụ:
* A, B, C, …
* Result, Output,Out, …
* \_X, \_x16, \_, …

1. **Sự kiện trong Prolog:**

* Sự kiện trong Prolog được viết thành một vị từ Prolog để máy có thể hiểu được.
* Ví dụ: Cho cây gia hệ như hình dưới, các nút chỉ người, còn các mũi tên chỉ quan hệ *cha mẹ* *của* (parent of):

**+** Sự kiện: *“Bình là cha mẹ của Phú”* được viết thành vị từ trong Prolog: *parent\_of(bình, phú)*

+ Sự kiện: *“Phú là cha mẹ của My”* được viết thành vị từ trong Prolog: *parent\_of(phú, my)*

+ Tương tự ta có các vị từ khác cũng được xây dựng theo quan hệ *parent* như sau:

* *parent\_of(an, phú)*
* *parent\_of(bình, tân)*
* *parent\_of(phú, hòa)*
* *parent\_of(hòa, nam)*

+ Sau khi hệ thống Prolog nhận được chương trình này, thực chất là một cơ sở  
dữ liệu, người ta có thể đặt ra các câu hỏi liên quan đến quan hệ *parent*. Ví dụ:

* ?- parent\_of(bình, phú).
* Prolog trả lời: *Yes* do có sự kiện *parent\_of(bình, phú)* tồn tại
* ?- parent\_of(phú, nam).
* Prolog trả lời: *No* do không có sự kiện *parent\_of(phú, nam)* tồn tại

1. **Luật trong Prolog:**

* Mỗi luật có hai thành phần:

+ Phần bên phải (RHS: Right Hand Side) chỉ *điều kiện*, còn được gọi là *thân*  
(body) của luật.

+ Phần bên trái (LH: Left Hand Side S) chỉ *kết luận*, còn được gọi là *đầu*  
(head) của luật.

* Luật được thêm vào cơ sở dữ liệu để vân dụng và chứng minh suy luận người dùng là đúng (True) hay sai (False).
* Luật là dạng tổng quát với các đối tượng bất kỳ, không có các đối tượng cụ thể. Ví dụ:

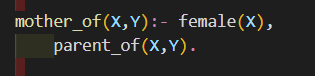
+ Ta định nghĩa quan hệ *mother* như sau:

woman

mother

paren

+ Từ định nghĩa trên ta có luật:



* Luật này hiểu là: (Với mọi X và Y, X là mẹ của Y nếu X là nữ và X là cha mẹ của Y)

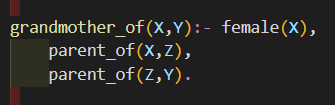
+ Ta có định nghĩa mối quan hệ *grandmother* như sau:

woman

parent grandmother

parent

+ Từ định nghĩa trên ta có luật:



* Luật này phát biểu là:

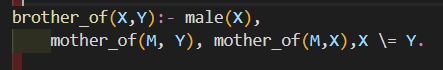
(Với mọi X và Y, X là bà của Y nếu X là nữ và tồn tại Z sao cho X là cha mẹ của Z và Z là cha mẹ của Y)

+ Ta có định nghĩa mối quan hệ *brother* như sau:

mother mother

brother

+ Từ định nghĩa trên ta có luật:



* Luật này phát biểu là:

(Với mọi X và Y, X là anh em của Y nếu X là nam và M là mẹ của Y và tồn tại M là mẹ của X và X khác Y)