Chương IV

LẬP TRÌNH LÔGIC

IV.1. GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH LÔGIC PROLOG

1. Mở đầu

Một trong những ***mục tiêu*** lớn và quan trọng của trí tuệ nhân tạo (TTNT) là *mô phỏng trên máy tính những tư duy và hành vi* của con người. Trước những năm 1960, khi chưa có nhóm các ngôn ngữ lập trình (NNLT) chuyên phục vụ cho lĩnh vực TTNT như các NNLT xử lý ký hiệu (LISP- LISt Processing) và lập trình lôgic (PROLOG – PROgramming in LOGic), những thành tựu lý thuyết trong TTNT chưa được ứng dụng rộng rãi trong thực tế. Chương này nhằm giới thiệu *Prolog*, một trong những NNLT thuộc họ này.

NNLT Prolog do Alain Colmerauer và nhóm cộng sự thiết kế ra vào năm 1972 tại đại học Marseille ở Pháp. ***Prolog*** được xây dựng trên ***cơ sở*** lý thuyết của *lôgic vị từ*, nhằm *biểu diễn, mô tả* các tri thức cùng các mối quan hệ cơ bản giữa chúng. Từ đó, khi đã *được trang bị sẵn* một cơ chế suy luận thông qua một ***môtơ suy diễn***, Prolog sẽ *sản sinh ra và kiểm tra nhiều mối quan hệ hệ quả khác*.

Khác với nhiều NNLT thủ tục cổ điển truyền thống, ***Prolog*** thuộc nhóm ***NNLT mô tả***. Để giải quyết một bài toán, đặc biệt là các bài toán xử lý ký hiệu trừu tượng và suy luận lôgic, ta chỉ cần *chọn cách biểu diễn phù hợp và mô tả các đối tượng cùng các tính chất và mối quan hệ cơ bản* đã biết giữa chúng. Thông qua môtơ suy diễn của Prolog, ta sẽ *nhận được các câu trả lời* liên quan đến các mối *quan hệ nhân quả và các kết luận liên quan* đến bài toán cần giải quyết. Khi tri thức đã biết về bài toán *không tăng thêm*, việc *trả lời các kết luận khác liên quan đến bài toán cần giải là hiển nhiên thông qua các câu hỏi thêm mà không cần phải lập trình lại* như các NNLT kiểu thủ tục. Chính vì vậy, NNLT Prolog thuộc kiểu ***hội thoại*** và một *chương trình nguồn* trong Prolog ***thường gọn hơn rất nhiều*** so với một chương trình nguồn của các NNLT thủ tục truyền thống khác như Pascal, C, … nhằm cùng giải quyết một bài toán chuyên biệt trong lĩnh vực TTNT.

* ***Ví dụ I.1***: Để mô tả các *sự kiện (fact)*: "Socrate và Tèo là người", *qui tắc lôgic (hoặc luật, rule)*: "hễ bất cứ ai là người thì người đó sẽ phải chết", ta dùng hai vị từ chính: LàNgười(Ai), Chết(Ai) như sau:

LàNgười("Socrate").

LàNgười("Tèo").

Chết(Ai) **:-** LàNgười(Ai). //hoặc: Chết(Ai) **if** LàNgười(Ai).

Dựa trên các tri thức vừa mô tả, ta có thể yêu cầu Prolog *trả lời các câu hỏi* liên quan sau:

- *Goal:* LàNgười("Socrate") → yes
- *Goal:* LàNgười("mèo") → no

- *Goal:* Chết("Socrate") → yes
- *Goal:* Chết("mèo") → no

hoặc:

- *Goal:* LàNgười(Ai) → Ai = "Socrate", Ai = "Tèo" (2 solutions)
- *Goal:* Chết(Ai) → Ai = "Socrate", Ai = "Tèo" (2 solutions)

Các ***khái niệm chính*** trong Prolog là: vị từ (predicate), đối tượng, biến, sự kiện (fact), qui tắc (rule), câu hỏi (đích, mục tiêu: Goal), danh sách. Vài ***kỹ thuật*** thường được dùng trong Prolog là: quay lui (mặc định là tìm kiếm theo chiều sâu trên cây suy diễn), đệ qui, lát cắt.

2. Vị từ, sự kiện, qui tắc, mục tiêu trong Prolog

a. Vị từ, đối tượng, hằng, biến

* Quan hệ giữa các đối tượng trong một bài toán, một mệnh đề được biểu diễn bằng các ***vị từ*** và các ***đối*** của vị từ biểu diễn các ***đối tượng*** thuộc *quan hệ* hay có *tính chất nào* đó.

Để ***chuyển một mệnh đề thông thường sang dạng vị từ*** (*predicate*) trong Prolog, người ta thường *chọn động từ chính trong phần vị ngữ làm tên vị từ* và các *thành phần khác làm các đối* tương ứng của vị từ.

- ***Phân loại*** các đối tượng trong Prolog: *hằng, biến* hay *đối tượng phức hợp (*là một loại *đối tượng của một vị từ mà bản thân nó lại là một vị từ khác)*.

- Các ***kiểu dữ liệu chuẩn*** (cơ sở) có sẵn trong Prolog: *int, real, char, string, symbol*. Ngoài ra, Prolog còn cho phép các *kiểu dữ liệu riêng do người dùng định nghĩa*.

- ***Kiểu (dữ liệu) của đối tượng*** có thể thuộc kiểu chuẩn hoặc do người dùng định nghĩa.

- Ta qui ước ***Name*** là một *dãy ký tự* (gồm: *chữ cái* la tinh, chữ *số* hoặc *dấu gạch dưới* "_") mà *ký tự đầu tiên phải là chữ cái*. (Do đó, trong *Name* không được có khoảng trắng " " và không được bắt đầu là chữ số !).

Khi dùng *Name* để biểu diễn tên của các đối tượng hay vị từ, người ta thường dùng một trong hai cách viết sau để *dễ đọc và tránh hiểu nhầm nghĩa*: ghế_dài, ghế_đẩu hay ghếDài, ghếĐẩu; suy_luận hay SuyLuận, so_1_Han_Thuyen hay so_1_HanThuyen, … (cho ví dụ vài dãy ký tự không phải là một *Name* ?)

- *Tên của đối tượng* là một *Name*.

- **Hằng** là đối tượng mà giá trị của chúng được gán ngay khi khai báo trong phần **constants** và *không đổi* trong suốt chương trình. ***Tên của hằng*** là một *Name* mà *ký tự đầu tiên phải là* ***chữ cái thường*** (đặc biệt là khi sử dụng giá trị của chúng).

- **Biến** là đối tượng mà giá trị của chúng có thể *thay đổi* trong chương trình. ***Tên của biến*** (*trừ biến vô danh*, được ký hiệu là *dấu gạch dưới* "_") là một *Name* mà ký tự đầu tiên là ***chữ cái hoa***. Biến được phân thành ***3 loại***:

. ***biến ràng buộc*** là biến mà tại thời điểm đang xét nó *được gán với một giá trị xác định*;

. ***biến tự do*** là biến mà tại thời điểm đang xét nó *chưa hoặc không được gán với một giá trị xác định nào cả*; do đó, tại những thời điểm khác nhau một biến có thể *chuyển từ tự do sang ràng buộc và ngược lại* trong quá trình quay lui khi tìm kiếm lời giải;

. ***biến vô danh*** (được ký hiệu là dấu gạch dưới "_") là biến mà ta *không quan tâm* (do đó, không cần lưu lại hay ghi nhớ) *đến giá trị* của nó.

Chú ý: Các *biến* trong Prolog khi được sử dụng *không cần phải khai báo trước* (vì thông thường, chúng đã được khai báo ngầm trong phần khai báo kiểu cho các vị từ)

b. Sự kiện, qui tắc:

Các vị từ trong Prolog thường được thể hiện dưới hai dạng: sự kiện hay qui tắc.

- ***Sự kiện (fact)*** là các *mệnh đề hằng đúng* và nó thường được thể hiện bằng *các vị từ mà các đối của chúng đều là đối tượng hằng*. Chẳng hạn, trong ví dụ 1, *LàNgười("Socrate")* là một sự kiện.

- ***Qui tắc (rule)*** là một *mệnh đề kéo theo đúng* và chúng thường được thể hiện bởi *các vị từ mà các đối của chúng chứa các biến*. Qui tắc thường ***gồm hai phần***: *PhầnĐầu* và *PhầnThânQuiTắc*, chúng được nối với nhau bởi từ khoá "***if***" hoặc "**:-**", theo ***cú pháp sau***:

PhầnĐầu **if** *PhầnThânQuiTắc***.**

hoặc:

PhầnĐầu **:-** *PhầnThânQuiTắc***.**

(*PhầnThânQuiTắc luôn được kết thúc* bởi một dấu chấm "**.**") Chẳng hạn, trong ví dụ 1, *Chết(Ai)* là một qui tắc.

. Để tạo nên các qui tắc phong phú, ta có thể dùng thêm trong *PhầnThânQuiTắc* các *phép toán lôgic* để nối kết các mệnh đề: **and** (hay dấu phẩy "**,**"), **or** (hay dấu chấm phẩy "**;**"), **not(vị từ)**.

. Các vị từ (sự kiện, qui tắc) phải được *khai báo kiểu* trong phần *Predicates* trước khi được *liệt kê, định nghĩa và được sử dụng* trong phần *Clauses* và *Goal*.

c. Mục tiêu, câu hỏi:

Sau khi lập trình, các yêu cầu của người dùng có thể được biết thông qua các *câu hỏi*, vị từ đưa vào trong phần mục tiêu *Goal*. Có hai loại *Goal*:

- ***Goal trong*** (đưa nội dung cần hỏi vào phần Goal trong chương trình nguồn): *chỉ tìm* ra *lời giải đầu tiên* và *không tự động xuất* chúng ra màn hình. Nếu muốn *xuất chúng*, có thể sử dụng vị từ chuẩn xuất dữ liệu: *write*; nếu muốn *tìm các lời giải kế tiếp*, có thể dùng thêm vị từ chuẩn luôn sai: *fail*.

- ***Goal ngoài***: *tìm* và *tự động xuất mọi lời giải* (ra cửa sổ hội thoại *Dialog*, với từ khóa Goal: ở đầu mỗi câu hỏi). Nếu chỉ muốn *xuất ra lời giải đầu tiên*, có thể sử dụng vị từ chuẩn *lát cắt* "**!**".

3. Cấu trúc chính của một chương trình trong Prolog

Một chương trình của Prolog thường bao gồm *một số các phần chính* và *theo thứ tự* sau:

Constants % Đoạn khai báo các đối tượng hằng
Domains % Đoạn khai báo kiểu các đối tượng riêng của người dùng
Database % Đoạn khai báo kiểu cho các vị từ của cơ sở dữ liệu
Predicates % Đoạn khai báo kiểu cho vị từ sẽ dùng trong các đoạn sau
Clauses % Đoạn liệt kê sự kiện và định nghĩa qui tắc đã khai báo trong phần *Predicates*
Goal % Đoạn đưa vào các vị từ câu hỏi, có thể đặt trước phần *Clauses*

Chú ý: Một chương trình có thể thiếu một số (hoặc tất cả !) phần trên.

a. Đoạn khai báo các đối tượng hằng: được bắt đầu bởi từ khoá *Constants* và theo cú pháp sau:

TênĐốiTượngHằng = TrịĐốiTượngHằng
[*Name*]

TrịĐốiTượngHằng có kiểu dữ liệu chuẩn (*int, real, char, string, symbol*).

b. Đoạn khai báo kiểu dữ liệu riêng của người dùng: được bắt đầu bởi từ khoá *Domains* và theo một trong các cú pháp sau:

. Dãy TênKiểuDữLiệu = *KiểuDữLiệu*
[*Name, ...*]

trong đó: *KiểuDữLiệu* là kiểu dữ liệu chuẩn (*int, real, char, string, symbol*) hoặc kiểu dữ liệu riêng của người dùng;

. Dãy TênKiểuDanhSách = KiểuPhầnTửChung*****
[*Name* **,** ...] [*KiểuDữLiệu*]
. TênKiểuĐốiTượngPhứcHợp = Dãy TênVịTừ(KiểuĐối, …)
[*Name*] [*Name(KiểuDữLiệu, ...)***;** …]
. **file** = Dãy TênFileHìnhThức
[*Name***;** ...]

Các tên file của các thiết bị chuẩn sau không cần phải khai báo: *screen, keyboard, printer*.

c. Đoạn khai báo kiểu cho vị từ sẽ dùng trong các đoạn *Goal* và *Clauses* ở phần tiếp theo được bắt đầu bởi từ khoá *Predicates* và theo cú pháp sau:

. TênVịTừ(DãyKiểuĐối)
[*Name*]([*KiểuDữLiệu, ...*])

Đoạn khai báo kiểu cho các vị từ của cơ sở dữ liệu (CSDL) được bắt đầu bởi từ khoá *Database* và cũng theo cú pháp như phần *Predicates*. Điểm khác biệt của các vị từ khai báo trong đoạn *Database* là: các vị từ tương ứng với chúng

trong phần Clauses có các *đối là các đối tượng hằng hoặc các biến ràng buộc* để có thể *đưa vào CSDL ở bộ nhớ trong* theo các vị từ chuẩn riêng biệt.

Chú ý: Có thể khai báo *nhiều vị từ cùng tên* nhưng: *cùng số đối với kiểu khác nhau* hoặc có *số đối khác nhau* hoặc thậm chí *không có đối nào* ! Khi đó, chúng phải được *khai báo liên tiếp nhau*.

d. Đoạn liệt kê các sự kiện và định nghĩa các qui tắc đã khai báo trong phần *Predicates* hoặc *Database*, được bắt đầu bởi từ khoá Clauses theo cú pháp sau:

. *Liệt kê các sự kiện*:

TênVịTừ(Dãy TrịHằng)**.**

. *Định nghĩa các qui tắc*:

TênVịTừ(Dãy ĐốiTượng) **:-** *PhầnThânQuiTắc*(Dãy ĐốiTượng)**.**

hoặc:

TênVịTừ(Dãy ĐốiTượng) **if** *PhầnThânQuiTắc*(Dãy ĐốiTượng)**.**

trong đó: *PhầnThânQuiTắc* có thể gồm các vị từ, mệnh đề *được nối kết với nhau bởi các phép toán lôgic* như: and (“,”), or (“;”), not(*VịTừ(Đối)*), nhưng *Đối* của *VịTừ* trong *not* phải là *hằng* hoặc *biến ràng buộc*. Tất cả các vị từ đều phải được kết thúc bởi một dấu chấm “**.**”.

* ***Chú ý***:

- Các *vị từ cùng tên* phải được liệt kê hay định nghĩa *liên tiếp nhau*.
- Cơ chế tìm kiếm mặc định trong Prolog theo chiều sâu. *Thứ tự* của các *mệnh đề cùng tên* (*cùng các đối của chúng*) trong phần *Clauses* có thể *có ý nghĩa khác nhau* và *ảnh hưởng đến tốc độ tìm kiếm*.
- Cần để ý đến *thứ tự và độ ưu tiên* các phép toán lôgic trong *PhầnThânQuiTắc*.
- Qui tắc:

A:- B1; B2.

tương đương với:

A:- B1**.**
A:- B2**.**

- Để biểu diễn mệnh đề: **B and (C or D)**, dùng qui tắc sau là sai:

A:- B, C; D. % hoặc: A:- B, (C;D). cũng sai !

Khi đó, có nhiều cách đúng để biểu diễn chúng:

C1: A:- B, E.
E:- C; D.

hoặc *C2*: A:- B, C.
A:- B, D.

hoặc *C3*: A:- B, C; B, D.

e.* Đoạn mục tiêu** (đối với ***Goal trong), để đưa vào các *vị từ câu hỏi*, kết hợp với các phép toán lôgic, được bắt đầu bởi từ khóa *Goal*.

Nếu nhìn chương trình theo quan niệm Top-Down, đoạn này có thể *đặt trước* phần *Clauses*.

***Ví dụ I.2** (minh họa việc dùng Goal trong, Goal ngoài; cơ chế tìm kiếm nghiệm bội của Goal ngoài)*:

Domains

Ai = symbol

Predicates

Thích(Ai, Ai)

Clauses

Thích("A", b).

Thích(b, c).

Thích(X, Z):- Thích(X, Y), Thích(Y, Z).

Goal %trong: *đưa vào trong chương trình nguồn*

%Các đối trong vị từ đều ràng buộc:

Thích("A", b). %Kết quả: rỗng

Thích("A", c), write("Đúng"). %Kết quả: Đúng

%Có đối trong vị từ là biến tự do:

Thích("A", Ai). %Kết quả: rỗng

Thích("A", Ai), **write("A thích: ", Ai, "; ")**.

%Kết quả (xuất một lời giải): A thích: b

Thích(Ai, c), write(Ai, " thích c; "), **fail**.

% Kết quả (xuất nhiều lời giải): b thích c; A thích c

Goal %ngoài: *chỉ đưa vào cửa sổ hội thoại*

%Các đối trong vị từ đều ràng buộc:

> Thích("A", b) → yes

%Có đối trong vị từ là biến tự do:

> Thích("A", Ai) → Ai=b, Ai=c (2 solutions).

> Thích("A", Ai), **!**. → Ai=b (1 solutions).

> Thích(Ai_1, Ai_2) → *kết quả là gì ? (Bài tập)*

***Ví dụ I.3**(minh họa: các phép toán lôgic trong PhầnThânQuiTắc, các qui tắc cùng tên)*:

Domains

Ai = symbol

Predicates

Me(Ai, Ai)

Cha(Ai, Ai)

Cha(Ai)

ChaMe(Ai, Ai, Ai)

ChaHoacMe(Ai, Ai)

Nam(Ai)

Nu(Ai)

Nguoi(Ai)

Dung()
Sai

Clauses

Cha(ba_1, con_1).
Cha(ba_1, con_2).
Cha(Ba):- Cha(Ba, _).

Me(ma_1, con_1).
Me(ma_2, con_2).

ChaMe(Ba, Ma, Con) :- Cha(Ba, Con), Me(Ma, Con).

ChaHoacMe(BaMa, Con) :- Cha(BaMa, Con) **;** Me(BaMa, Con).

Nam(con_1).
Nam(Ai):- Cha(Ai).

Nguoi(Ai):- Cha(Ai) **;** Me(Ai) **;** ChaHoacMe(_, Ai).

Nu(Ai):- Nguoi(**Ai**), **not**(Nam(**Ai**)).
% Nu(Ai):- ***not***(Nam(Ai)). % *không trả lời được câu hỏi*: Nu(Ai) !?

Dung(). %Hoac: Dung():- true.
Sai:- not(Dung()).

Goal %ngoài: Cho kết quả là gì với từng goal sau ?
> Nu(con_2) → ?
> Nu(Ai) → ?

**** Ví dụ I.4****(minh họa đối tượng phức hợp; cơ chế tìm kiếm nghiệm bội của Goal ngoài)*:

Domains

Ai, Ten = symbol
Gi = Mèo(Ten) ; Bồ(Ten) ; Rỗng

Predicates

Thich(Ai, Gi)

Clauses

Thich(a, Meo(miu)).
Thich(tu,rong).
Thich(b, Meo(miu)). Thich(b, Bo(miu)).
Thich(c, X) :- Thich(a, X) , Thich(b, X).

Goal %ngoài
> Thich(c, Gi) → Cho kết quả là gì ? (*Bài tập*)

IV.2. DANH SÁCH, ĐỆ QUI, LÁT CẮT TRONG PROLOG

Danh sách là một trong những cấu trúc dữ liệu rất quan trọng và được sử dụng thường xuyên trong *lập trình xử lý ký hiệu* bằng Prolog. *Đệ qui* là kỹ thuật thường dùng trong Prolog. Do *cơ chế tìm kiếm tự động mặc định trong Prolog theo chiều sâu*, nên, trong nhiều tình huống, nếu muốn *ngăn chặn* việc tìm kiếm tiếp tục không cần thiết, Prolog cho phép dùng *lát cắt* để thực hiện việc này.

1. Danh sách

a. Định nghĩa: Danh sách là một *dãy có thứ tự* các phần tử. Một cách *đệ qui*, ta có thể định nghĩa danh sách là:

- *rỗng* (không có phần tử nào), hoặc:
- gồm *một phần tử đầu và danh sách đuôi*

b. Khai báo, biểu diễn danh sách:

TênKiểuDanhSách = KiểuChungCácPhầnTử*

trong đó: *KiểuChungCácPhầnTử* có thể là kiểu dữ liệu chuẩn, kiểu do người dùng định nghĩa, kiểu phức hợp, hoặc thậm chí là danh sách.

- Danh sách *rỗng*: **[]**
- *Biểu diễn danh sách* theo *kiểu đệ qui*:

  BiếnKiểuDS = [**PhầnTử**Đầu | **DanhSách**Đuôi]
- Biểu diễn *danh sách* theo *kiểu liệt kê*:

  [**PhầnTử**ThứNhất **, PhầnTử**ThứHai | **DSách**CònLại]

Về mặt lý thuyết, có thể biểu diễn danh sách bằng *vị từ hoặc cây, biểu đồ ngang*.

*** Ví dụ II.1**: Tìm phần tử đầu và danh sách đuôi của các danh sách sau:
[a, b, c], [a], [[1, 2], [3], [4,5]] , [] ? (*Bài tập*)

2. Đệ qui - Cơ chế quay lui và tìm kiếm nghiệm bội trong Prolog

a. Cơ chế quay lui và tìm kiếm nghiệm bội: Minh họa qua ví dụ:

*** Ví dụ II.2**:

```
%trace QHệ_1_2
QHệ_1(a, b).
QHệ_1(a, c).

QHệ_2(d, b).
QHệ_2(e, c).
QHệ_2(d, c).
```

QHệ_2(e, b).

QHệ_1_2(B, M, C) :- QHệ_1(B, C), QHệ_2(M, C).

Goal % ngoài
> QHệ_1_2(a, e, Ai) → Ai=b, Ai = c

b. *Định nghĩa*: Một ***thao tác đệ qui F(x) trên tập D*** gồm 2 phần :
. *Điều kiện dừng*: Một thao tác sơ cấp (đã biết cách thực hiện trực tiếp) trên tập con $X_o \subset D$
. Một *lời gọi đệ qui F(H(x))*, với $H(x) \in D$, sao cho sau một số hữu hạn lần gọi đệ qui sẽ phải dẫn đến điều kiện dừng ($H(H(\ldots H()\ldots)) \in X_o$, $\forall x \in D \backslash X_o$).

**** Ví dụ II.3**(thao tác đệ qui thiếu điều kiện dừng)*:
Ba(B, C):- Con(C, B).
Con(C, B):- Ba(B, C).
Goal: > Ba(a,b) → ?

c. *Kỹ thuật khử đuôi đệ qui bằng cách dùng biến phụ*:
**** Ví dụ II.4***: Tính *chiều dài* của một danh sách. Do *một vị từ có thể có nhiều tác dụng*, nên chú thích thêm *dòng vào-ra (I, o)* để chỉ các cách thức sử dụng vị từ cho đúng.

Domains
ptu = real
ds = ptu*
i = integer
Predicates
ChieuDai(ds,i) % (i, i) – (i, o) *– (o, i), ! – (o, o), readchar(_)*
ChieuDaiPhu(ds, i, i) % *dòng vào-ra*: *(i, i, i) – (i, i, o)*
% ChieuDaiPhu(i,0,_): luôn khởi tạo biến thứ 2 là 0
Clauses
ChieuDai([], 0).
ChieuDai([_|Duoi], Kq) :- ChieuDai(Duoi,KqPhu), Kq = KqPhu + 1.

ChieuDaiPhu([], Kq, Kq).
ChieuDaiPhu([_|Duoi], Phu, Kq) :- PhuMoi=Phu+1,
ChieuDaiPhu(Duoi, PhuMoi, Kq).
Goal %ngoài
> ChieuDai([1,2,3], 3) → ? > ChieuDai([1,2,3], Kq) → ?
> ChieuDaiPhu([1,2,3], 0, 3) → ?
> ChieuDaiPhu([1,2,3], 0, Kq) → ?
> ChieuDaiPhu(Nao, 0, 1) → ?

Bài tập: Viết 2 qui tắc đệ qui (có và không có đệ qui đuôi) tính giai thừa của một số tự nhiên.

Chú ý: *Thứ tự các vị từ cùng tên* trong phần Clauses có thể *có ý nghĩa khác nhau*, *đặc biệt là các vị từ đệ qui*. Chẳng hạn, xem tác dụng của 2 cách viết qui tắc kiểm tra danh sách có 1 phần tử sau (và sửa lại cho đúng: *bài tập*):

Cách 1:

```
DS1PTu([_|D]) :-  DS1PTu(D).
DS1PTu([]).
```

Cách 2:

```
DS1PTu([]).
DS1PTu([_|D]) :-  DS1PTu(D).
```

với cách hỏi: DS1PTu(Nào) ?

d. Vài thao tác đệ qui cơ bản trên danh sách:

*** Ví dụ II.5***: Lập qui tắc kiểm tra *quan hệ thuộc giữa một phần tử và một danh sách*: Thuoc(Ptu, DSach) % (i, i) – (o, i) – (i, o) ?

```
Thuoc(PT, [PT|_]).
Thuoc(PT, [_|DSDuoi]) :- Thuộc(PT, DSDuoi).
```

*** Ví dụ II.6***: Lập qui tắc *nối hai danh sách*: Noi(DSach, DSach, DSach) % (i, i, i) – (i, i, o) – (o, o, i) – (i, o, i) – (o, i, i) ?

```
Noi([], L, L).
Noi([PT|L1], L2, [PT|L3]) :- Noi(L1, L2, L3).

ThuocMoi(PT, DS) :- Noi(_, [PT|_], DS).
```

3. Lát cắt (!)

Lý do dùng lát cắt: Do cơ chế tìm kiếm tự động mặc định trong Prolog theo *chiều sâu*, nhưng trong nhiều bài toán với các đặc trưng riêng, ta *không cần cơ chế quay lui* hoặc *tìm tiếp lời giải kế tiếp*. Khi đó, ta có thể dùng lát cắt để đạt được mục đích đó.

a. Cú pháp, tác dụng của lát cắt: Dùng dấu chấm than "**!**" để biểu diễn lát cắt. Lát cắt là *vị từ luôn đúng* và có tác dụng ***ngăn***: *cơ chế quay lui và tiếp tục tìm kiếm lời giải đối với các vị từ cùng tên với những vị từ đứng trước lát cắt* trong một mệnh đề nào đó. Điều đó dẫn đến việc *tiết kiệm thời gian lẫn bộ nhớ* khi dùng lát cắt thích hợp.

Ta minh họa lát cắt qua vài trường hợp sau:

- Đối với qui tắc không đệ qui:

ViTừ_1 :- a, !, b. **% (1)**

ViTừ_1 :- c. **% (2)**

Khi a sai thì vị từ c được thực hiện. Khi *a đúng*, sẽ *gặp "!" và thực hiện tiếp b (kể cả nghiệm bội của b, nếu b đúng)*, nhưng *sẽ không bao giờ tìm tiếp lời giải cho a và (kể cả trường hợp b sai) không quay xuống ViTừ_1 thứ hai trong (2) để kiểm tra c* , với các trị trong đối của ViTừ_1 tương ứng với lần gọi đó.

- Đối với qui tắc đệ qui:

ViTừ_2.

ViTừ_2 :- a, !, ViTừ_2. **% (3)**

ViTừ_2 :- b. **% (4)**

Khi a sai, (4) sẽ được kiểm tra. Khi *a đúng, ViTừ_2 trong (4) sẽ không được kiểm tra* ứng với lần gọi đó, *mặc dù ViTừ_2 trong thân qui tắc của (3) là sai* và *sẽ không bao giờ tìm tiếp lời giải cho a* .

b. Các tình huống có thể sử dụng lát cắt:

α- *Chỉ tìm lời giải đầu tiên ở Goal ngoài.*

β- Trong các bài toán *chỉ có duy nhất một lời giải*, *sau khi tìm được lời giải đầu tiên*, ta dùng lát cắt để *dừng ngay* quá trình tìm kiếm.

δ- Ta muốn *dừng khi một mệnh đề sai* (nhưng không *không cần tìm cách khác để thỏa mãn mệnh đề*), hoặc *thay cho vị từ not*, bằng cách sử dụng *tổ hợp lát cắt* và *vị từ fail* phù hợp.

γ- Dùng lát cắt trong các chương trình "*Sinh và Thử*".

**** Ví dụ II.7** (minh họa vài tình huống dùng lát cắt)*:

```
Domains
        i, ptu = integer
Predicates
        RangBuoc(ptu)  % minh họa tình huống δ
        GiaiThua(i, i) % minh họa tình huống β
        ThuongNguyen(i, i, i) % hai đối đầu ∈N, minh họa tình huống γ
        Sinh_1_So(i)
Clauses
    % RangBuoc(X) :- not(free(X)). % sai khi X là biến không ràng buộc
        RangBuoc(X) :- free(X), !, fail.
        RangBuoc(_).

        GiaiThua(0, 1) :- !.
        GiaiThua(N, Kq) :- N1 = N-1, GiaiThua(N1, Kq1), Kq = Kq1*N.

        ThuongNguyen(B, C, T) :- Sinh_1_So(T), T*C<=B, (T+1)*C>B, !.
```

```
Sinh_1_So(0).
Sinh_1_So(Sau) :- Sinh_1_So(Truoc), Sau = Truoc + 1.
```

.

Ví dụ II.8 *(Bài tập: tìm các lỗi sai, hạn chế và mở rộng các trường hợp có dòng vào-ra tốt hơn)*: Tính số cha và mẹ của mỗi người.

```
Domains
    Ai = symbol
    i = integer
Predicates
    SoChaMe(Ai, i)
Clauses
    SoChaMe("Adam", 0).
    SoChaMe("Eva", 0).
    SoChaMe(_,2).
% Gợi ý: SoChaMe("Adam", 0): !. hoặc: SoChaMe("Adam", BN): !, BN=0., …
```

Chú ý:

- Phép toán **X = Y** có thể hiểu theo hai nghĩa:
  . Phép ***so sánh*** nếu cả X và Y là biến (hay biểu thức) *ràng buộc*;
  . Phép ***gán*** nếu chỉ *một trong hai vế là biến tự do*, còn vế kia là biểu thức hay biến ràng buộc (và sẽ vô nghĩa, bị lỗi khi cả hai vế đều không ràng buộc!)
- Có thể thay phép toán = bởi vị từ *Bằng(X,X)*.

Gợi ý tạo các lệnh cấu trúc như các NNLT cổ điển:

- if (ĐK) then A;
* else B;*

```
Neu(DK) :- !, A.
Neu( _ ) :- B.
```

- switch (BT)
* { case a1: A1; break;*
*   case a2: A2; break;*
*   …*
*   case an: An; break;*
*   default: B; break;*
* };*

```
%Gọi sử dụng: Switch(BT)
Switch(a1) :- !, A1.
Switch(a2) :- !, A2.
        …
Switch(an) :- !, An.
Switch( _ ) :- B.
```

- repeat A until (DK);

```
Repeat :- Lap, A, DK, !.

Lap.
Lap :- Lap.
```

- while (ĐK) do A;

```
While :- ĐK, !, A, While.
While.
```

- for (i=Bdau; i<=Kthuc; i = i+Step) do A(i);

```
% Gọi sử dụng: For(Bdau, Kthuc, Step)
For(I, N, Step) :- Dấu(Step, D), D*(I-N) > 0, !.
 For(I, N, Step) :- A(i), I_Sau = I+Step, For(I_Sau, N, Step).
```

IV.3. CÁC VÍ DỤ

1. Bài toán "Tháp Hà Nội" (Minh họa vị từ đệ qui)

Ví dụ III.1: Có 3 cọc A, B, C. Trên cọc A có n đĩa tròn chồng lên nhau, với đường kính khác nhau, sao cho đĩa bé luôn nằm trên đĩa lớn; hai cọc B, C trống. Tìm cách chuyển n đĩa từ A đến C với số lần di chuyển đĩa ít nhất (2^n-1), sao cho: mỗi lần chỉ di chuyển 1 đĩa bé nhất trên cùng của một cọc tùy ý sang một cọc khác để đĩa bé luôn nằm trên đĩa lớn.

```
%Program Thap_HaNoi
Domains
        i  = integer
        s = symbol
Predicates
        HaNoi(i, i)
        ChuyenDia(i, i, s, s, s, i)
Clauses
        HaNoi(N, SoLanDiChuyen) :- ChuyenDia(N, N, a, c, b, SoLanDiChuyen).

        ChuyenDia(1, Nhan, Tu, Den, _, 1) :- !, nl,
                write("Chuyen dia: ", Nhan, " tu ", Tu, " den ", Den).
        ChuyenDia(N, Nhan, Tu, Den, Tgian, Tong):-
                N1=N-1, NhanPhu=Nhan-1,
                ChuyenDia(N1, NhanPhu, Tu, TGian, Den, Tong1),
                ChuyenDia(1, Nhan, Tu, Den, Tgian, 1),
                ChuyenDia(N1, NhanPhu, TGian, Den, Tu, Tong2),
```

```
        Tong = Tong1 + Tong2 + 1.
Goal %trong
    makewindow(1, 3, 7, "Thap Ha Noi", 0, 0, 25, 80),
    write("Nhap so dia n (1<=n<=12): "), readint(N), HaNoi(N, Tong),
    write("Tong so lan di chuyen dia: ", Tong)..
```

Kết quả của chương trình với n=2 là:

```
        Chuyen dia: 1 tu a den b
        Chuyen dia: 2 tu a den c
        Chuyen dia: 1 tu b den c
        Tong so lan di chuyen dia: 3
```

2. Bài toán xử lý vi phân ký hiệu *(Minh họa bài toán xử lý ký hiệu)*:

Tính đạo hàm của một hàm số theo một biến bất kỳ.

```
%Program Xử lý vi phân ký hiệu
Domains
    KyHieu = symbol
    So     = real
    BieuThuc = bien(KyHieu); hang(So);
               cong(BieuThuc,BieuThuc); tru(BieuThuc,BieuThuc);
               nhan(BieuThuc,BieuThuc); chia(BieuThuc,BieuThuc);
               cos(BieuThuc); sin(BieuThuc);
               tg(BieuThuc); cotg(BieuThuc);
               ln(BieuThuc); log(BieuThuc,BieuThuc);
               exp(BieuThuc); mu(BieuThuc,BieuThuc);
               luyThua(BieuThuc,BieuThuc)
Predicates
    d(BieuThuc,KyHieu,BieuThuc)
Clauses
    d(hang(_),_,hang(0)).
    d(bien(X),X,hang(1)):-!.
    d(bien(_),_,hang(0)).

    d(cong(U,V),X,cong(U1,V1)):- d(U,X,U1), d(V,X,V1).
    d(tru(U,V),X,tru(U1,V1)):- d(U,X,U1), d(V,X,V1).
    d(nhan(U,V),X,cong(nhan(U1,V),nhan(U,V1))):- d(U,X,U1), d(V,X,V1).
    d(chia(U,V),X,chia(tru(nhan(U1,V),nhan(U,V1)),nhan(V,V))):-
        d(U,X,U1), d(V,X,V1).

    d(sin(U),X,nhan(cos(U),U1)) :- d(U,X,U1).
    d(cos(U),X,tru(hang(0),nhan(sin(U),U1))) :- d(U,X,U1).
    d(tg(U),X,chia(U1, mu(cos(U),hang(2)))) :- d(U,X,U1).
```

```
d(cotg(U),X,nhan(hang(-1),chia(U1, mu(sin(U),hang(2))))) :- d(U,X,U1).

d(ln(U),X,chia(U1,U)):-d(U,X,U1).
d(log(U,V),X, Kq) :- d(chia(ln(V),ln(U)),X,Kq).

d(exp(U),X,nhan(exp(U),U1)) :- d(U,X,U1).
d(mu(U,V),X,nhan(mu(U,V),W)):- d(nhan(V,ln(U)),X,W).

d(luyThua(U,hang(A)),X,nhan(hang(A),nhan(U1,luyThua(U,hang(A_1))))):-
        A_1 = A-1, d(U,X,U1).
Goal %ngoài: tính vi phan của x.sin(x)
    d(nhan(bien(x),sin(bien(x))),x, Kq).
```

Kết quả (*chưa rút gọn*) của chương trình là:

```
Kq=cong(nhan(hang(1),sin(bien(x))),nhan(bien(x),
                                    nhan(cos(bien(x)),hang(1))))
```

3. Bài toán suy luận lôgic (Minh họa lập trình cho bài toán lập luận lôgic)

Ví dụ III.2: Trên một đảo nọ, chỉ có hai nhóm người: một nhóm chỉ gồm những người luôn nói thật và nhóm còn lại chỉ gồm những người luôn nói láo. Một nhà thông thái đến đảo này và gặp ba người A, B, C. A nói: "Trong ba người chúng tôi chỉ có đúng một người nói thật", C nói: "Cả ba người chúng tôi đều nói láo". Hỏi trong ba người A, B, C, ai nói thật, ai nói láo ?

Ta *biểu diễn mỗi phương án* của bài toán bằng danh sách [A, B, C], sao cho mỗi phần tử chỉ gồm một trong hai trạng thái 0, 1, tương ứng với trạng thái nói lá hoặc nói thật. Ta *mô tả câu nói của A và C* như sau: nếu A nói thật thì Tổng = A+B+C = 1, nếu A nói láo thì Tổng ≠ 1; nếu C nói thật thì Tổng = A+B+C = 0, nếu C nói láo thì Tổng > 0.

```
%Program Noi lao, noi that
Domains
    i, ptu = integer
    ds  = ptu*
Predicates
    TaoDS(i,ds)
    Quet(ptu)
    Giai(i,ds)
    SuKien(symbol,ds)
    Tong(ds,integer)
    Viet(ds,i)
Clauses
    Quet(0).
    Quet(1).
```

```
TaoDS(N,[]) :- N <= 0, !.
TaoDS(N,[Dau|Duoi]) :- Quet(Dau),N1 = N - 1, TaoDS(N1,Duoi).

Giai(N,DS) :- TaoDS(N,DS), SuKien("A",DS), SuKien("C",DS).

SuKien("A",DS) :- DS = [1,_,_], Tong(DS,1).
SuKien("A",DS) :- DS = [0,_,_], Tong(DS,So), So <> 1.

SuKien("C",DS) :- DS = [_,_,1], Tong(DS,0).  % phi li !
SuKien("C",DS) :- DS = [_,_,0], Tong(DS,So), So > 0.

Tong([],0) :- !.
Tong([Dau|Duoi],Kq) :- Tong(Duoi,KqDuoi), Kq = KqDuoi + Dau.

Viet([],_) :- nl,!.
Viet([1|D],I):-!,I1=I+1,I2 =I1+64,char_int(C,I2), write(C," noi that\n"),
              Viet(D,I1).
Viet([0|D],I):-!,I1=I+1,I2 =I1+64,char_int(C,I2), write(C," noi lao\n"),
              Viet(D,I1).
```

Goal *% trong*

```
Giai(3,DS),Viet(DS,0),nl.
```

Kết quả của chương trình là:

A noi that
B noi lao
C noi lao

IV.4. PHỤ LỤC: VÀI VỊ TỪ CHUẨN CỦA PROLOG

I. NHẬP VÀ XUẤT DỮ LIỆU

Các vị từ xuất và nhập dữ liệu xét ở đây sẽ được thao tác từ các thiết bị vào ra hiện thời, ngầm định là màn hình và bàn phím.

1. Các vị từ nhập dữ liệu

Các ký hiệu sẽ sử dụng trong biểu diễn cú pháp các vị từ có ý nghĩa như sau: Sau tên vị từ các biến có các kiểu được liệt kê tiếp theo. Sau đó, dòng vào ra đối với các biến của vị từ sẽ được chỉ ra, trong đó các biến có vị trí tương ứng với ký tự:

. 'o' phải là biến tự do (sau khi vị từ được thực hiện các kết qủa của nó sẽ được gán vào các biến này);

. 'i' phải là biến ràng buộc hoặc đối tượng hằng (chúng cung cấp dữ liệu ban đầu để các vị từ này thực hiện).

a) **readln(StringVariable)**, (string) – (o): trong đó StringVariable là biến tự do có kiểu string. Vị từ này có tác dụng đọc một chuỗi từ thiết bị vào hiện thời (và tất nhiên được gán cho biến StringVariable) cho đến khi gặp ký tự xuống dòng (phím Enter, có mã ASCII tương ứng là '\13') được ấn.

b) **readint(IntgVariable)**, (integer) – (o): đọc một số nguyên từ thiết bị vào hiện thời cho đến khi gặp ký tự xuống dòng được ấn.

c) **readreal(RealVariable)**, (real) – (o): đọc một số thực từ thiết bị vào hiện thời cho đến khi gặp ký tự xuống dòng được ấn.

d) **readchar(CharVariable)**, (char) – (o): đọc một ký tự (nhưng không xuất hiện ký tự trên màn hình) từ thiết bị vào hiện thời và không cần kết thúc bằng ký tự xuống dòng.

e) **file_str(DosFileName, StringVariable)**, (string, string) – (i, o) (i, i): chuyển đổi nội dung văn bản giữa file có tên thực là DosFileName và biến chuỗi có tên StringVariable (cho đến khi gặp ký tự kết thúc file eof, tương ứng với tổ hợp phím Ctl-z hay mã ASCII là '\26').

f) **inkey(CharVariable)**, (char) – (o): đọc một ký tự từ bàn phím (gán cho biến CharVariable). Vị từ này chỉ đúng khi có một ký tự được ấn từ bàn phím.

g) **keypressed**: kiểm tra việc một phím được bấm hay chưa nhưng không đọc ký tự này vào. Vị từ này chỉ đúng khi có một ký tự được ấn từ bàn phím.

h) **keypressed**: kiểm tra việc một phím được bấm hay chưa nhưng không đọc ký tự này vào. Vị từ này chỉ đúng khi có một ký tự được ấn từ bàn phím.

2. Các vị từ xuất dữ liệu

a) **write(Variable|Constant *)**: trong đó dấu * chỉ ra rằng các biến Variable (ràng buộc) hay hằng Constant có thể được lặp lại và chúng được viết cách nhau bằng các dấu phẩy ‘,’. Vị từ này có tác dụng xuất đến thiết bị ra một số đối tượng.

b) **nl**: xuống dòng.

c) **writef(FormatString, Variable|Constant *)**: xuất ra một số đối có định khuôn dạng theo cú pháp sau: %-m_p$, trong đó:

. ‘-’: canh lề bên trái, nếu không có dấu ‘-’ sẽ là canh lề bên phải;

. ‘m’: xác định độ rộng vùng nhỏ nhất cho đối;

. ‘p’: xác định độ chính xác các số sau dấn chấm thập phân hay số lớn nhất các ký tự được in ra của chuỗi;

. ‘$’ là một trong các ký hiệu quy ước chuẩn:

d: số thập phân dạng chuẩn.

x: số thập lục phân.

s: chuỗi (symbols hay strings).

c: ký tự (chars hay integers).

g: số thực dạng ngắn nhất ngầm định.

e: số thực dạng mũ.

f: số thực dạng dấu phẩy động.

d) **“\n”**: ký tự xuống dòng.

“\t”: ký tự lùi vào một số khoảng trắng.

“\nnn”: ký tự đưa vào bảng mã ASCII tương ứng.

II. CÁC THAO TÁC FILE

a) **openread(SymbolicFileName, DosFileName)**, (file, string) – (i, i): mở một file trên đĩa DosFileName để đọc và gán cho tên file hình thức (trong phần khai báo kiểu file) SymbolicFileName.

b) **openwrite(SymbolicFileName, DosFileName)**, (file, string) – (i, i): mở một file trên đĩa để ghi và gán cho tên file hình thức; nếu file đó đã tồn tại thì nó sẽ được xóa trước khi gọi.

c) **openappend(SymbolicFileName, DosFileName)**, (file, string) – (i, i):mở một file trên đĩa để nối thêm dữ liệu và gán cho tên file hình thức.

d) **openmodify(SymbolicFileName, DosFileName)**, (file, string) – (i, i):mở một file trên đĩa để sửa đổi (đọc và ghi) và gán cho tên file hình thức. Khi

dùng vị từ này, nên kết hợp với vị từ filepos để cập nhật file tại vị trí tùy chọn.

e) **readdevice(SymbolicFileName)**, (file) – (i) (o): đặt lại hay hiển thị thiết bị đọc vào file đang mở (để đọc hay sửa đổi), ngầm định là bàn phím.

f) **writedevice(SymbolicFileName)**, (file) – (i) (o): đặt lại hay hiển thị thiết bị ghi vào file đang mở (để đọc ghi hoặc nối thêm hay sửa đổi), ngầm định là màn hình.

g) **closefile(SymbolicFileName)**, (file) – (i): đóng file (dù nó đã mở hay chưa).

h) **filepos(SymbolicFileName, FilePosition, Mode)**, (file, real, integer) – (i, i, i) (i, o, i): đưa con trỏ đến hay trả lại giá trị của vị trí FilePosition trong file tính từ chế độ Mode: 0: đầu file, 1: vị trí hiện tại trong file, 2: cuối file.

i) **eof(SymbolicFileName)**, (file) – (i): kiểm tra con trỏ có ở cuối file hay không. Vị từ này đúng nếu con trỏ ở vị trí cuối file.

j) **existfile(DosFileName)**, (string) – (i): vị từ này thành công nếu file có tên chỉ ra tồn tại trong thư mục hiện thời.

k) **deletefile(DosFileName)**, (string) – (i): xóa file có tên chỉ ra.

l) **renamefile(OldDosFileName, NewDosFileName)**, (string, string) – (i, i): đổi tên file cũ thành tên file mới.

m) **disk(DosPath)**, (string) – (i) (o): thiết lập hay trả lại ổ đĩa và đường tìm kiếm thư mục mặc định.

III. CÁC THAO TÁC ĐỐI VỚI MÀN HÌNH VÀ CỬA SỔ

1. *Màn hình*

Màu nền	**Giá trị**	**Màu chữ**	**Giá trị**
Black	0	Black	0
Blue	16	Blue	1
Green	32	Green	2
Cyan	48	Cyan	3
Red	64	Red	4
Magenta	80	Magenta	5
Brown	96	Brown	6
White	112	White	7

		Grey	8
		Light Blue	9
		Light Green	10
		Light Cyan	11
		Light Red	12
		Light Magenta	13
		Yellow	14
		White (High Intensity)	15

a) **attribute(Attr)**, (integer) – (i) (o): cho (đặt hay trả lại) giá trị thuộc tính của màn hình. Giá trị đặc trưng cho thuộc tính của màn hình, dựa vào bảng trên đây, được tính như sau:
   1. Chọn một màu chữ và một màu nền;
   2. Cộng những giá trị nguyên tương ứng với màu trong bảng 1;
   3. Cộng thêm giá trị đó với 128 nếu muốn các chữ hiển thị nhấp nháy.

b) **scr_char(Row, Column, Char)**, (integer, integer, char) – (i, i, i) (i, i, o): cho ký tự Char trên màn hình tại vị trí có tọa độ (Row, Column).

c) **scr_attr(Row, Column, Char)**, (integer, integer, char) – (i, i, i) (i, i, o): cho thuộc tính của màn hình tại vị trí có tọa độ (Row, Column).

d) **filed_str(Row, Column, Length, String)**, (integer, integer, integer, string) – (i, i, i, i) (i, i, i, o): cho chuỗi String trên màn hình tại một vùng bắt đầu ở vị trí có tọa độ (Row, Column) và có độ dài là Length ký tự.

e) **filed_attr(Row, Column, Length, Attr)**, (integer, integer, integer, integer) – (i, i, i, i) (i, i, i, o): cho thuộc tính của màn hình tại một vùng bắt đầu ở vị trí có tọa độ (Row, Column) và có độ dài là Length ký tự.

f) **cursor(Row, Column)**, (integer, integer) – (i, i) (i, o): dịch con trỏ đến vị trí có tọa độ (Row, Column) hay trả lại tọa độ của con trỏ hiện thời.

2. Hệ thống cửa sổ

a) **makewindow(WindowNo, ScrAtt, FrameAtt, FrameStr, Row, Column, Height, Width)**, (integer, integer, integer, string, integer, integer, integer, integer) – (i, i, i, i, i, i, i, i) (o, o, o, o, o, o, o, o): xác định một vùng màn hình làm cửa sổ.

b) **makewindow(WindowNo, ScrAtt, FrameAtt, FrameStr, Row, Column, Height, Width, ClearWindow, FrameStrPos, BorderChars)**, (integer,

integer, integer, string, integer, integer, integer, integer, integer, integer, string) – (i, i, i, i, i, i, i, i, i) (o, o, o, o, o, o, o, o, o, o): xác định một vùng màn hình làm cửa sổ với các thuộc tính sau:

ClearWindow	= 0	không xóa cửa sổ sau khi tạo,
	= 1	xóa cửa sổ sau khi tạo;
FrameStrPos	= 255	canh tít ở giữa,
	<> 255	đặt tít tại vị trí FrameStrPos;
BorderChars	một chuỗi gồm 6 ký tự để vẽ khung:	
	ký tự thứ 1: góc trên bên trái,	
	ký tự thứ 2: góc trên bên phải,	
	ký tự thứ 3: góc dưới bên trái,	
	ký tự thứ 4: góc dưới bên phải,	
	ký tự thứ 5: đường kẻ ngang,	
	ký tự thứ 6: đường kẻ dọc.	

c) **shiftwindow(WindowNo),** (integer) – (i) (o): đổi cửa sổ làm việc đến cửa sổ thứ WindowNo (vẫn giữ nguyên trạng thái của cửa sổ trước đó) hay trả lại số của cửa sổ đang làm việc.

d) **gotowindow(WindowNo),** (integer) – (i): di chuyển nhanh đến cửa sổ thứ WindowNo mà không lưu nội dung của cửa sổ cũ vào bộ đệm cửa sổ.

e) **resizewindow**: điều chỉnh lại kích thước cửa sổ như kiểu thực đơn.

f) **resizewindow(StartRow, NoOfRows, StartCol, NoOfCols)**, (integer, integer, integer, integer) – (i, i, i, i): điều chỉnh lại kích thước cửa sổ một cách trực tiếp.

g) **colorsetup(Main_Frame)**, (integer) – (i): thay đổi màu cửa sổ hay khung tùy theo giá trị của:

Main_Frame	= 0	đổi màu cửa sổ
	= 1	đổi màu khung

h) **existwindow(WindowNo)**, (integer) – (i): vị từ này thành công nếu tồn tại cửa sổ số WindowNo.

i) **removewindow**: loại bỏ cửa sổ hiện thời.

j) **removewindow(WindowNo, Refresh)**, (integer, integer) – (i, i): loại bỏ cửa sổ số WindowNo với chế độ xóa nền hay không xóa nền (nếu Refresh bằng 1 hoặc 0).

k) **clearwindow**: xóa cửa sổ hiện thời về màu nền của nó.

l) **window_str(ScreenString)**, (string) – (i) (o): chuyển đổi nội dung giữa cửa sổ hiện thời và chuỗi ScreenString.

m) **window_at(Attribute)**, (integer) – (i): xác định thuộc tính cho cửa sổ hiện thời.

n) **scroll(NoOfRows, NoOfCols)**, (integer, integer) – (i, i): cuộn nội dung của lên (hay xuống) X=NoOfRows dòng tùy theo nó > 0 (hay < 0) và sang trái (hay phải) X=NoOfCols cột tùy theo nó > 0 (hay < 0).

o) **framewindow(FrameAttr)**, (integer) – (i): thay đổi thuộc tính của khung cửa sổ.

p) **framewindow(FrameAttr, FrameStr, FrameStrPos, FrameTypeStr)**, (integer, string, integer, integer) – (i, i, i, i): thay đổi thuộc tính của khung cửa sổ theo các chế độ sau:

FrameAttr	= thuộc tính khung
FrameStr	= tít của khung
FrameStrPos	= nhận giá trị từ 0 đến độ rộng cửa sổ. Nó đặt vị trí cho tít trên khung. Tít sẽ đặt ở giữa nếu nó bằng 255
FrameTypeStr	= một chuỗi gồm 6 ký tự để chỉ dạng của khung.

IV. CÁC VỊ TỪ CHUẨN KHÁC

1. Thao tác chuỗi

a) **frontchar(String, FrontChar, RestString)**, (string, char, string) – (i, o, o) (i, i, o) (i, o, i) (i, i, i) (o, i, i): tách một chuỗi thành ký tự đầu và chuỗi còn lại.

b) **fronttoken(String, Token, RestString)**, (string, string, string) – (i, o, o) (i, i, o) (i, o, i) (i, i, i) (o, i, i): tách một chuỗi thành một 'từ' và chuỗi còn lại.

c) **frontstr(Length, InpString, StartString, RestString)**, (integer, string, string, string) – (i, i, o, o): tách từ một cho trước InpString ra một chuỗi con StartString bắt đầu từ ký tự thứ nhất có độ dài là Length và chuỗi còn lại RestString.

d) **concat(String1, String2, String3)**, (string, string, string) – (i, i, o) (i, o, i) (o, i, i) (i, i, i): nối String1 và String2 thành String3: String3 = String1 + String2.

e) **strlen(String, Length)**, (string, integer) – (i, i) (i, o) (o, i): chiều dài của String là Length. Với dòng vào ra (o, i) ta được một chuỗi gồm Length ký tự trắng.

f) **isname(StringParam)**, (string) – (i): kiểm tra chuỗi có là một 'name' (một dãy các ký tự chữ hoặc số hoặc dấu '_' nhưng bắt đầu bằng chữ hay ký tự '_' trong Turbo Prolog không.

2. *Chuyển đổi kiểu*

a) **char_int(CharParam, IntgParam)**, (char, integer) – (i, o) (o, i) (i, i): chuyển đổi giữa một ký tự và mã ASCII tương ứng với nó.

b) **str_int(StringParam, IntgParam)**, (string, integer) – (i, o) (o, i) (i, i): chuyển đổi giữa một chuỗi và một số nguyên tương ứng với nó.

c) **str_char(StringParam, CharParam)**, (string, char) – (i, o) (o, i) (i, i): chuyển đổi giữa một chuỗi và một ký tự tương ứng với nó.

d) **str_real(StringParam, RealParam)**, (string, real) – (i, o) (o, i) (i, i): chuyển đổi giữa một chuỗi và một số thực tương ứng với nó.

e) **upper_lower(StringInUpperCase, StringInLowerCase)**, (string, string) – (i, o) (o, i) (i, i): chuyển đổi giữa một chuỗi ký tự hoa và một chuỗi ký tự thường.

f) **upper_lower(CharInUpperCase, CharInLowerCase)**, (char, char) – (i, o) (o, i) (i, i): chuyển đổi giữa một ký tự hoa và một ký tự thường.

3. *Thao tác đối với cơ sở dữ liệu trong*

a) **consult(DosFileName)**, (string) – (i): gọi ra và bổ sung một file cơ sở dữ liệu trong (file đó được khởi tạo và lưu một cơ sở dữ liệu trong không tên bằng vị từ save(?)).

b) **consult(DosFileName, InternalDatabaseName)**, (string, InternalDatabaseName) – (i, i): gọi ra và bổ sung một file cơ sở dữ liệu trong (file đó được khởi tạo và lưu một cơ sở dữ liệu trong không tên bằng vị từ save(?, ?)).

c) **save(DosFileName)**, (string) – (i): lưu mọi mệnh đề của cơ sở dữ liệu trong không tên.

d) **save(DosFileName, InternalDatabaseName)**, (string, InternalDatabaseName) – (i, i): lưu mọi mệnh đề của cơ sở dữ liệu trong có tên.

e) **assert(Term)**, (InternalDatabaseDomain) – (i) hay **asserta(Term)**, (InternalDatabaseDomain) – (i): chèn một sự kiện vào đuôi một cơ sở dữ liệu trong không tên.

f) **assertz(Term)**, (InternalDatabaseDomain) – (i): chèn một sự kiện vào đuôi một cơ sở dữ liệu trong không tên.

g) **retract(Term)**, (InternalDatabaseDomain) – (_): vị từ không tất định (nondeterm) này loại bỏ sự kiện chỉ ra của cơ sở dữ liệu trong không tên.

h) **retract(Term, InternalDatabaseDomain)**, (InternalDatabaseDomain) – (_, i): vị từ không tất định (nondeterm) này loại bỏ sự kiện chỉ ra của cơ sở dữ liệu trong có tên.

i) **retractall(Term)**, (InternalDatabaseDomain) – (_): vị này loại bỏ mọi sự kiện của cơ sở dữ liệu trong không tên. Nên sử dụng biến vô danh trong vị từ này. Vị từ này không bao giờ sai.

j) **retractall(_, InternalDatabaseDomain)**, (_, InternalDatabaseDomain) – (_, i): vị này loại bỏ mọi sự kiện của cơ sở dữ liệu trong có tên. Nên sử dụng biến vô danh trong vị từ này.

4. Các lệnh liên quan đến hệ điều hành DOS

a) **system(DosCommandString)**, (string) – (i): thực hiện một lệnh của hệ điều hành DOS trong môi trường Turbo Prolog.

b) **dir(Path, Filespec, Filename)**, (string, string, string) – (i, i, o): xem các files trong thư mục và chọn một file bằng menu để xem nội dung.

c) **exit**: thoát ra một chương trình và trở về môi trường làm việc của Prolog.

5. Các lệnh linh tinh khác

a) **random(RealVariable)**, (real) – (o): tạo một số ngẫu nhiên thực trong khoảng [0, 1).

b) **random(MaxValue, RandomInt)**, (integer, integer) – (i, o): tạo một số nguyên ngẫu nhiên trong khoảng [0, RandomInt).

c) **sound(Duration, Frequency)**, (integer, integer) – (i, i): tạo ra âm thanh có trường độ Duration và tần số Frequency.

d) **beep**: tạo ra một tiếng còi ngắn.

e) **date(Year, Month, Day)**, (integer, integer, integer) – (o, o, o) (i, i, i): cho ngày, tháng, năm của đồng hồ hệ thống.

f) **time(Hours, Minutes, Seconds, Hundredths)**, (integer, integer, integer, integer) – (o, o, o, o) (i, i, i, i): cho giờ của đồng hồ hệ thống.

g) **trace(on/off)**, (string) – (o) (i): đặt chế độ lần vết hay không cho chương trình.

h) **findall(Variable, Atom, ListVariable)**: ghi các giá trị của biến trong Atom trùng tên với Variable vào danh sách ListVariable.

i) **not(Atom)**: vị từ này có giá trị logic phủ định của vị từ Atom. Lưu ý: không được dùng biến tự do trong vị từ Atom.

j) **free(Variable)**: vị từ này chỉ đúng khi Variable là biến tự do.

k) **bound(Variable)**: vị từ này chỉ đúng khi Variable là biến ràng buộc.

l) **fail**: vị từ này luôn luôn sai.

m) **true**: vị từ này luôn luôn đúng.

n) **Các phép toán số học trong Turbo Prolog**: +, -, *, /, mod, div.

o) **Các phép toán quan hệ trong Turbo Prolog** (có giá trị không chỉ đối với các số mà còn đối với các ký tự, chuỗi, symbol): >, <, =, >=, <=, <>, ><.

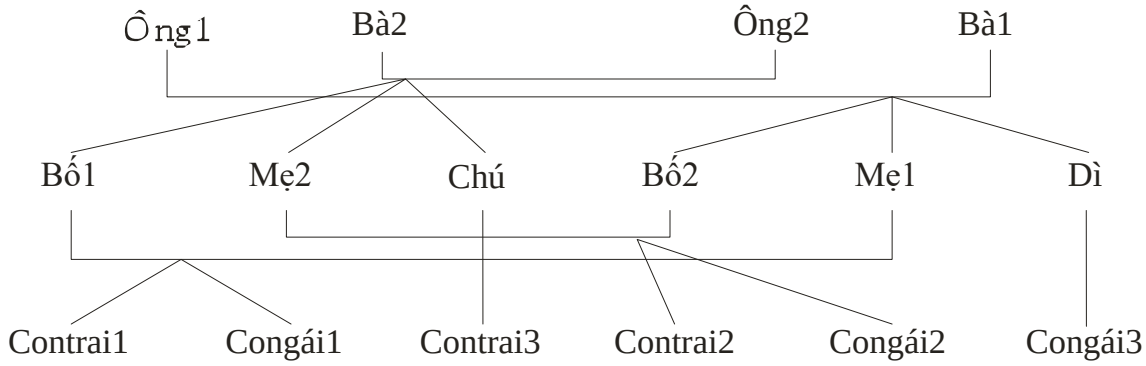
p) **Các hàm trong Turbo Prolog**: sin, cos, tan, arctan, ln, log, exp, sqrt, round (số nguyên làm tròn), trunc (số nguyên chặt cụt), abs.

BÀI TẬP CHƯƠNG IV

(Phần bài tập làm quen với vị từ trong Prolog)

Bài 1:

Xét cây gia phả sau:



Cây được mô tả bởi các vị từ quan hệ sau: cha(Cha, Con), mẹ(Mẹ, Con), nam(DO), nữ (DB).

Hãy liệt kê các quan hệ còn lại như: anhEm(Ng1, Ng2), con(Con, Cha, Mẹ), chú(Chú, Cháu), bà(Bà, Cháu), anhEmHọ(Ng1, Ng2)... từ những vị từ trên.

Suy ra mối quan hệ giữa hai người bất kỳ.

Bài 2:

Một dịch vụ du lịch phục vụ những chuyến đi trong 1 tuần hoặc 2 tuần ở Rome, London, Tunis.

Trong một quyển sách giới thiệu về dịch vụ du lịch, với mỗi địa điểm có giá cả đi lại (tùy vào thời gian của chuyến đi) và chi phí cho 1 tuần, tùy vào địa điểm và mức độ tiện nghi: khách sạn, nhà trọ, cắm trại ngoài trời.

a) Nhập vào các sự kiện mô tả quyển sách trên (giá cả do bạn nhập vào).
b) Hãy mô tả quan hệ ChuyếnĐi(TP, SốTuần, TiệnNghi, ChiPhí).
c) Mô tả quan hệ ChuyếnĐi_ÍtTốnKém(TP, SốTuần, TiệnNghi, ChiPhí, ChiPhíMax) sao cho chi phí của chuyến đi thấp hơn ChiPhíMax.

Bài 3:

Trong một dịch vụ tư vấn hôn nhân, liệt kê một danh sách các cặp xứng đôi có thể làm quen theo các nhận định sau:

- nam(n, t, c, a), nữ(n, t, c, a).
  n: tên; t: chiều cao.
  c: màu tóc (vàng, hung, nâu, hạt dẻ).
  a: tuổi (trẻ, trung niên, già).

- SởThích(n, m, l, s) (sở thích) có nghĩa là: Người n thích loại nhạc m (classique, pop, jazz), thích văn học loại l (phiêu lưu, văn học viễn tưởng, trinh thám), và chơi môn thể thao s(quần_vợt, bơi, chạy bộ).
- tìm(n, t, c, a): Người n tìm bạn có các tính chất t, c, a.

Hai người x và y gọi là hợp nhau nếu có cùng ngoại hình (t, c, a) và sở thích về: l, m, s.

a) Nhập dữ liệu về các khách hàng.
b) Mô tả qui tắc hợp_ngoạihình(x, y) và cùng_sở thích(x, y).
c) Viết chương trình xác định những cặp xứng đôi.

(Phần bài tập về danh sách, đệ qui, lát cắt)

Bài 4:

Xây dựng các vị từ đệ quy (có hoặc không sử dụng nhát cắt và sửa đổi phần đuôi của đệ quy, nếu có thể) để:

a) Tìm (hoặc xóa) phần tử thứ 1, thứ 2, phần tử cuối cùng trong một danh sách.
b) Tìm (hoặc lấy ra) phần tử đầu tiên trong danh sách trùng với một giá trị cho trước.
c) Tìm và lấy ra tất cả các phần tử trong danh sách trùng với một giá trị cho trước.
d) Tính giá trị trung bình cộng của một danh sách kiểu số thực.
e) Xuất ra tất cả các cặp phần tử kế tiếp nhau của danh sách.
f) Tạo danh sách tất cả các phần tử ở vị trí chẵn của danh sách dữ liệu L:
   - Giữ nguyên trật tự xuất hiện trong L.
   - Đảo ngược trật tự.
g) Viết chương trình trộn(u, v, w) nhận xen kẽ từng phần của u và v để tạo thành danh sách w.
h) Tạo danh sách chứa các số chẵn của một danh sách số nguyên cho trước.
i) Đảo ngược trật tự mọi phần tử của một danh sách.
j) Xuất ra mọi hoán vị của những phần tử của danh sách cho trước.
k) Xây dựng vị tự Thuộc trong đó có thêm một đối để trả về trị TRUE nếu phần tử x thuộc danh sách l cho trước và FALSE trong trường hợp ngược lại.
l) Xây dựng qui tắc NằmNgoài(x, L) để kiểm tra xem x không thuộc danh sách L.
m) Xây dựng qui tắc KhácNhau(L) để kiểm tra danh sách L không có phần tử trùng nhau.
n) Xây dựng qui tắc PhầnĐầu(m, L) để kiểm tra danh sách m là phần đầu của danh sách L.
o) Xây dựng qui tắc NằmTrongLiênTiếp(m, L) để kiểm tra mọi phần tử của m đều nằm liên tiếp trong danh sách L.

p) Xây dựng qui tắc PhânHoạch(L, u, v, w) để phân hoạch danh sách L thành 3 danh sách u, v, w.
q) Thêm một phần tử vào cuối danh sách.
r) Tìm và xóa phần tử thứ k của danh sách l cho trước.
s) Chèn một phần tử x sau phần tử thứ k của danh sách L (0 <= k <= length(L)).
t) Xây dựng danh sách mà các phần tử là căn bậc hai của các phần tử tương ứng trong danh sách L các số thực cho trước.

Bài 5:

Tìm n bộ 3 số tự nhiên (x, y, z) đầu tiên thỏa tính chất Pythagore sau:

$0 < x < y < z$

$x^2 + y^2 = z^2$

UCLN(x, y, 1): Ước chung lớn nhất của x và y bằng 1.

Bài 6 (+):

Tìm n_k số tự nhiên đầu tiên $\overline{a_1, a_2, ..., a_n}$ thỏa:

$$\overline{a_1.a_2.a_n} = \sum_{i=1}^{n} a_i^k \qquad (k = 3, 4, 5)$$

VD: k = 3, $153 = 1^3 + 5^3 + 3^3$.

Bài 7:

Kiểm tra số tự nhiên N có phải là số nguyên tố không? Tìm n số nguyên tố đầu tiên.

Bài 8:

Tìm các số nguyên tố <= n (sàng Eratosthene).

Bài 9:

a) Nhập 2 danh sách số nguyên dương không có phần tử trùng lặp từ bàn phím.
b) Xây dựng các phép toán và các quan hệ trên tập hợp: Hội, Giao, Hiệu, Thuộc, Bao hàm.

Bài 10:

Cài đặt các thuật toán sắp xếp: chọn, chèn, tráo đổi đơn giản và sắp xếp nhanh.

Bài 11 (+):

Các đỉnh của một đồ thị được đánh số từ 0 -> n. Mỗi cung của đồ thị tương ứng với qui tắc Cung(i, j) nối gốc i với ngọn j.

Xây dựng qui tắc ĐườngĐi(x, y, L) với L là danh sách biểu diễn đường đi không lặp từ x đến y.

Bài 12 (+):

Cài đặt cây nhị phân tìm kiếm và các thao tác cơ bản.

Bài 13:

Giải quyết bài toán "Tháp Hà Nội" bằng đệ quy.

Bài 14 (*): (8 hậu)

Đặt 8 con hậu lên bàn cờ sao cho không con nào ăn con nào.

Bài 15 (*): (Dominos)

Có một danh sách các quân cờ đôminô. Mỗi quân cờ đôminô được biểu diễn bởi Cặp(i, j) ứng với 2 con số của nó. Người ta tìm cách sắp xếp chúng theo tính chất sau: Mỗi con đôminô có thể xếp vào một trong hai đầu của hàng đã xếp.

Xuất phát từ một con đôminô trong số các con được cho, hãy xây dựng một dãy hợp lệ chứa tất cả các đôminô.

Bài 16 (*):

Giải quyết bài toán ma phương cấp n lẻ.

(Phần bài tập về trò chơi, suy luận lôgic)

Bài 17: (Ngựa vằn)

Có 5 người quốc tịch khác nhau sống trong 5 căn nhà màu khác nhau. Mỗi người có một con thú riêng, một thức uống riêng, và hút thuốc của những hãng khác nhau.

- Người Anh sống trong căn nhà màu đỏ.
- Con chó thuộc về người Tây Ban Nha.
- Người uống cà phê trong căn nhà màu xanh lá cây.
- Người Ukrain uống trà.
- Ngôi nhà xanh lá cây ở cạnh ngôi nhà trắng, bên phải.
- Người hút thuốc OldGold nuôi những con sên.
- Người uống sữa ở ngôi nhà chính giữa.
- Người hút thuốc Kool sống trong căn nhà màu vàng.
- Người Nauy sống trong căn nhà đầu tiên bên trái.
- Người hút thuốc Chesterfield sống cạnh người có nuôi cáo.
- Người hút thuốc Kool sống cạnh người có nuôi ngựa.
- Người hút thuốc Gitanes uống rượu.
- Người Nhật hút thuốc Craven
- Người Nauy sống cạnh ngôi nhà màu xanh da trời.

Hỏi: Ai uống nước? Ai là chủ con ngựa vằn?

Bài 18 (*): (Trò chơi tốt đen và tốt trắng)

Bắt đầu từ 3 tốt đen và 3 tốt trắng đã được sắp xếp và cách nhau bởi một khoảng trắng.

VD: BBB_NNN.

Mỗi lần đánh chỉ có 1 trong 4 phép dịch chuyển sau được thi hành:

Tốt đen trượt sang trái:

BNB_NNB -> BNBN_NB

Tốt đen nhảy sang trái:

BN_BNNB -> BNNB_NB

Tốt trắng trượt sang phải:

BNB_NNB -> BN_BNNB

Tốt trắng nhảy sang phải:

BBN_NNB -> B_NBNNB

Viết 1 chương trình để chuyển từ trạng thái đầu sang trạng thái kết (mọi tốt trắng đều nằm bên phải).

Bài 19 (*):

Cho 1 biểu thức với phép * (Nhan) và + (Cong). Hãy sửa đổi để được 1 tổng các tích bằng cách thêm vào dấu ngoặc.

VD: a * (b + c) + (d + e) * (f + g) = (((((a * b + a * c) + d * f) + e * f) + d * g) + e * g)

Có nghĩa là:

Cong(Nhan(a, Cong(b, c), Nhan(Cong(d, e), Cong(f, g)))) cho ra Cong(Cong(Cong(Cong(Cong(Nhan(a, b), Nhan(a, c)), Nhan(d, f)), Nhan(e, f)), Nhan(d, g)), Nhan(e, g))

Hãy xây dựng chương trình trên theo 2 giai đoạn:

- Phân phối Nhan sao cho không có Nhan nào có Cong làm đối.
- Chuyển thành cây sao cho nhánh phải của Cong không chứa Cong khác, nhánh phải của Nhan không chứa Nhan khác.

Bài 20 (*):

Cho biểu thức dạng And(p, q), Or(p, q), Not(q), trong đó p, q là biến hoặc là biểu thức cùng dạng. Ta có:

And(Or(p, q), r) <=> Or(And(p, r), And(q, r))
And(p, Or(q, r)) <=> Or(And(p, q), And(p, r))
Not(Or(p, q)) <=> And(Not(p), Not(q))
Not(And(p, q)) <=> Or(not(p), not(q))
Not(Not(p)) <=> p

Hãy chuyển biểu thức theo yêu cầu sau:

Mỗi nút Or chỉ có tổ tiên là Or.
Mỗi nút And chỉ có tổ tiên là And.

a) Viết chương trình thực hiện công việc trên.
b) Thay đổi chương trình để mỗi nhánh trái Or không chứa nút Or khác, và mỗi nhánh trái And không chứa And khác.

Bài 21 (*): (Những ông chồng hay ghen)

Có ba cặp vơ chồng phải qua sông nhưng chỉ có một chiếc thuyền chỉ chở được tối đa hai người. Hãy lập trình cho các chuyến đi để sao cho không có bà vợ nào ở trên bờ hay trên thuyền với những người đàn ông X khác mà không có chồng mình hoặc không có đồng thời với vợ của X.

Bài 22 (*): (Sói, dê và bắp cải)

Có một con sói, một con dê và một bắp cải phải qua sông. Chỉ có một người chèo thuyền, chỉ chở được mỗi lần một trong số chúng. Nếu không có mặt người chèo thuyền thì dê sẽ ăn bắp cải và sói sẽ ăn thịt dê. Hãy lập trình cho các chuyến đi.

TÀI LIỆU THAM KHẢO

[1] Bạch Hưng Khang, Hoàng Kiếm. *Trí tuệ nhân tạo - Các phương pháp và ứng dụng*. NXB Khoa học Kỹ thuật, 1989.
[2] Hoàng Kiếm. *Giải một bài toán trên máy tính như thế nào (tập 1 và 2).* NXB Giáo dục, 2001.
[3] Nguyễn Thanh Thủy. *Trí tuệ nhân tạo - Các phương pháp giải quyết vấn đề và kỹ thuật xử lý tri thức.* NXB Giáo dục, 1995.
[4] A. Thayse. *Approche logique de l'intelligence artificielle (T1, T2, T3).* Dunod, Paris, 1990.
[5]. Ivan Bratko, *Prolog - Programming for artificial intelligence*, Addison-Wesley, 1990.