Prolog

```
alunos('Lucas Tavares', 'Igor Sousa Silva').
oi :- write("hello world!").
```

Prolog é uma linguagem lógica, estática, e é usada principalmente em Inteligência Artificial (Processamento de linguagem natural) - pela sua facilidade de implementação em buscas de banco de dados.

Sintaxe do Prolog

Termos:

- Átomos (string): `elephant`, `b`, `abcXYZ`, `x_123`, `como_voce_esta_hoje`, `'Este tambem e um átomo do Prolog.'` (obs: dependendo do sistema prolog que esteja usando, pode ou não aceitar caracteres especiais como + - * = < > : & nos átomos)
- Números: todos os sistemas prologs usam inteiros, podendo ser precedido por um -. Alguns sistemas aceitam floats.
- Variáveis: `X`, `Elefante`, `_4711`, `X_1_2`, `MinhaVariavel`, `_` (variável anônima)
- Termos Compostos: `e_maior(horse, X).`, `f(g(X, _), 7).`

Cláusulas, programas e queries.

- Programas em prolog: fatos e regras -> cláusulas -> sequências de cláusulas.
- Fatos: são um predicado seguido de um .
  - `bigger(whale, _).`
  - `life_is_beautiful.`
- Regras: é um predicado com o cabeça, e o corpo - um conjunto de predicados separados por vírgula.
  - `is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).`

Basico

```
bigger(elephant, horse).
bigger(horse, donkey).
bigger(donkey, dog).
bigger(donkey, monkey).
```

?- bigger(horse,donkey).
true.

```
bigger(elephant, horse).
bigger(horse, donkey).
bigger(donkey, dog).
bigger(donkey, monkey).

is_bigger(X, Y) :- bigger(X, Y).
is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).
```

?- bigger(elephant,monkey).
false.

?- is_bigger(elephant,monkey).
true .

saida

```
bigger(elephant, horse).
bigger(horse, donkey).
bigger(donkey, dog).
bigger(donkey, monkey).

is_bigger(X, Y) :- bigger(X, Y).
is_bigger(X, Y) :- bigger(X, Z),
is_bigger(Z, Y).
```

?- is_bigger(X,donkey).
X = horse ;
X = elephant ;
false.

```
aluno(uranos,10,8).
aluno(gaia,8,9).
aluno(zeus,2,4).

situacao(X,R):- aluno(X,Z,Y),
(Z+Y)/2 >=7, R=passou.
situacao(X,R):- aluno(X,Z,Y),
(Z+Y)/2 <7, R=reprovou.
```

?- situacao(gaia,X).
X = passou .
?- situacao(zeus,X).
X = reprovou.

comparação prolog vs c

```prolog
fatorial(1, 1). % base case
fatorial(N, Resultado) :- % recursion step
N > 1,
N1 is N - 1,
fatorial(N1, Result1),
Resultado is Result1 * N.
```

```cpp
#include <iostream>
using namespace std;
long long int fatorial(long long int p){
    if (p==0)
        return 1;
    else{
        return p*fatorial(p-1);
    }
}
int main(){
    long long int n;
    while (true){
        cin >> n;
        cout << "fatorial:" << fatorial(n) << endl;
    }
    return 0;
}
```

exemplo de expressividade em prolog

```c
//c++
struct familia{
  string* pais;
  string* filhos;
  int qpais,qfilhos;
};
struct criou{
  struct familia* relacao;
  int qrelacao;
}Deuses;
void criafamilia(int np,int nf,int geracao){
  Deuses.relacao[geracao].qfilhos=nf;
  Deuses.relacao[geracao].qpais=np;
  Deuses.relacao[geracao].pais=(string*)malloc(np*sizeof(string));
  Deuses.relacao[geracao].filhos=(string*)malloc(nf*sizeof(string));
}
void insereparente(int op,string deus,int at,int geracao){
  if(op==1){
      Deuses.relacao[geracao].pais[at]=deus;
  }
  else{
      Deuses.relacao[geracao].filhos[at]=deus;
  }
}
void criarvore(){
  Deuses.relacao=(struct familia*)
        malloc(Deuses.qrelacao*sizeof(struct familia));
}
```

```
/*Arvore genealogica dos deuses da mitologia
grega em prolog*/
/*CRIADORES*/
criou(chaos,
  [tartaro, gaia, eros, nix, erebo,
anteros]).
criou(gaia, uranos).
criou([gaia, uranos],
  [ciclopes, cronos, reia, febe, ceos,
  hiperiao, japeto, oceano, tetis,
equidna]).
criou([reia,cronos],
  [hestia, demeter, hera, hades, poseidon,
zeus] ).
criou([zeus,demeter],[persefone, zagreu]).
criou([zeus,hera],
  [ares, ilitia, eris, hebe, hefesto,
angelo]).
criou([zeus,leto],[apolo, artemis]).
```

```cpp
//c++
bool gerou(string p,string f){
  int i,j,k;
  for(i=0;i<Deuses.qrelacao;i++){
      for(j=0;j<Deuses.relacao[i].qpais;j++){
          if(Deuses.relacao[i].pais[j] == p){
              for(k=0;k<Deuses.relacao[i].qfilhos;k++){
                  if(Deuses.relacao[i].filhos[k] == f){
                      return true;
                  }
              }
          }
      }
  }
  return false;
}
```

```
------
opçoes
0-termina
1-gerou
2-parente
1
pai e filho
chaos gaia
1
```

```
------
opçoes
0-termina
1-gerou
2-parente
1
pai e filho
gaia zeus
0
```

?- criou(chaos,gaia).
false.

?- gerou(chaos,gaia).
true.

?- gerou(gaia,zeus).
false.

```prolog
/* prolog
gerou(Parente, Descendente) :-
   criou(Parente, Lista_Descendente),
   member(Descendente, Lista_Descendente).
gerou(Parente, Descendente) :-
   criou(Lista_Parente, Lista_Descendente),
   member(Parente, Lista_Parente),
   member(Descendente, Lista_Descendente).
gerou(X, Y) :- criou(X,Y).
```

Obs:

```
------
opçoes
0-termina
1-gerou
2-parente
1
pai e filho
chaos X
0
```

?- gerou(chaos, X).
X = tartaro ;
X = gaia ;
X = eros ;
X = nix ;
X = erebo ;
X = anteros ;
X = [tartaro, gaia, eros, nix, erebo, anteros].

```c++
//c++
bool parente(string p1,string p2){
  int i,j,k;
  if(gerou(p1,p2)){
      return true;
  }
  else{
    for(i=0;i<Deuses.qrelacao;i++){
      for(j=0;j<Deuses.relacao[i].qpais;j++){
        if(Deuses.relacao[i].pais[j]==p1){
          for(k=0;k<Deuses.relacao[i].qfilhos;k++){
            if(gerou(p1,Deuses.relacao[i].filhos[k]) && parente(Deuses.relacao[i].filhos[k],p2)){
                      return true;
            }
          }
        }
      }
    }
  }
  return false;
}
```

```prolog
/*prolog
parente(Parente, Descendente) :-
   gerou(Parente, Descendente).
parente(X, Y) :-
gerou(X,Z),parente(Z,Y).
```

---
opçoes
0-termina
1-gerou
2-parente
2
parente1 e parente2
tartaro gaia
0

---
opçoes
0-termina
1-gerou
2-parente
2
parente1 e parente2
gaia zeus
1

---
opçoes
0-termina
1-gerou
2-parente
2
parente1 e parente2
demeter X
0

?- parente(gaia,zeus).
true .

?- parente(tartaro,gaia).
false.

?- parente(demeter,X).
X = persefone ;
X = zagreu ;
false.

Bibliografias:

Qual a diferença entre uma linguagem estática e dinâmica?

Árvore Genealógica dos deuses

An Introduction to Prolog Programming, by Ulle Endriss, University of Amsterdam

Swish-Prolog sistema e como utilizá-lo

História e origens