Relatório Técnico sobre a Linguagem de Programação Prolog e suas Tecnologias: Testes e Implementações utilizando o SWI Prolog

Víctor Macêdo Carvalho[1] | E-mail: [1]victmacc@ufpi.eu.br

I. Resumo

Contexto: O Prolog é uma linguagem funcional intimamente ligada com a lógica matemática, tornando-a uma ferramenta indispensável para o estudo de lógica. Sua estrutura conceitual é diferente de outras linguagens de programação, o que pode tormá-la um impasse para leigos alguns dos programadores.

Problema: Este relatório tem como objetivo detalhar os algoritmos utilizados, bem como a descrição de chamadas de sistemas e tecnologias utilizadas para responder alguns problemas que podem serem resolvidos com a linguagem Prolog. No entanto, a compreensão do Prolog pode ser um obstáculo para alguns leitores.

Resultados: Este projeto apresenta uma seção que reforça a informação já desenvolvida, bem como apresenta resumidamente o experimento, para mitigar algumas sentenças, como, por exemplo, o que foi aprendido com a realização dos experimentos, e quais foram os resultados do experimento. Isso ajuda a tornar a compreensão da linguagem mais fácil para programadores iniciantes.
Palavras-chave: Prolog; experimentos; resultados.

1. Introdução

Este trabalho tem como objetivo apresentar uma abordagem para a resolução de questões que avaliam a linguagem de programação Prolog, por meio de testes e implementações realizados com a ferramenta SWI Prolog. A linguagem Prolog é reconhecida por sua estrutura conceitual diferente das linguagens de programação convencionais, sendo uma linguagem funcional baseada em lógica matemática. Nesse sentido, compreender a linguagem Prolog e avaliar sua eficácia na resolução de problemas é fundamental para o desenvolvimento da área de programação e para o avanço das soluções tecnológicas.

A compreensão da linguagem Prolog pode ser um desafio para programadores que não estão familiarizados com sua estrutura conceitual. Além disso, pode ser difícil avaliar a eficácia da linguagem na resolução de problemas sem a realização de testes e experimentos.

O trabalho apresenta seções específicas que fomentam o desenvolvimento do projeto por meio de figuras que abordam as partes do código (cláusulas) e também o retrato do exercício em questão. Essas informações são seguidas de experimentos realizados no SWI Prolog, que permitem avaliar a eficácia da linguagem na resolução de problema.

A conclusão do trabalho apresenta uma avaliação geral do projeto e do desempenho da linguagem Prolog na resolução de problemas. Os resultados obtidos por meio dos experimentos permitem compreender melhor a estrutura conceitual da linguagem e sua eficácia em diferentes situações. Além disso, o trabalho permite verificar a importância da realização de testes e experimentos para a avaliação e compreensão de uma linguagem de programação. Esses resultados contribuem para o avanço da área de programação e para o desenvolvimento de soluções tecnológicas cada vez mais eficientes.

2. Seções Específicas

Nesta seção, serão apresentados os aspectos funcionais do programa, com detalhamento dos algoritmos empregados, descrições de chamadas de sistemas e tecnologias utilizadas quando necessário.

2.1. Realização da 1º questão

A Figura 1 mostra a contextualização da questão, e, por conseguinte, o que a questão solicita para com a solução, mostrando dados e informações a respeito do problema em questão.

1) Crie uma base de dados fixa chamada CALÇADOS em Prolog contendo as seguintes informações por calçados: CÓDIGO, Nome, Tipo(chinelo, sandália, sapato, tênis), número, cor, marca e preço e que tenha um menu com as seguintes opções:

   I. Mostrar os dados de um determinado calçado, dado o CÓDIGO;
   II. Mostrar todos os calçados de um determinado tipo e tamanho;
   III. Mostrar todos os calçados de um determinado tipo e cor;
   IV. Mostrar todos os calçados de um determinado tamanho e cor.

Figura 1. Primeira questão

A Figura 2 é um predicado em Prolog que armazena informações sobre calçados (como nome, tipo, tamanho, cor, marca e preço) e permite ao usuário pesquisar e exibir esses calçados com base em várias opções.

As linhas de 01 – 14 são onde estão armazenados os fatos sobre cada calçado. Cada linha contém informações sobre um único calçado, com cada detalhe separado por vírgulas e delimitado por parênteses, a base de dados.

As linhas de 17 - 21 tem a cláusula que é usada para imprimir as informações de um calçado na tela. Cada argumento representa uma propriedade do calçado. A linha começa com write, que é usado para escrever uma mensagem na tela. O nl é usado para pular uma linha após cada propriedade.

As linhas de 27 - 38 tem a cláusula menu que é usada para exibir as opções disponíveis para o usuário. Cada opção é descrita com write, e a opção escolhida é lida com read.

As linhas de 40 - 75 são as cláusulas mostrar_calcado_codigo/1, mostrar_calcado_tipo_tamanho/2, mostrar_calcado_tipo_cor/2 e mostrar_calcado_tamanho_cor/2 são usadas para pesquisar e exibir calçados com base em várias opções.

As linhas de 105 – 114 são a cláusula escolher_opcao/1 é usada para direcionar o usuário para a opção escolhida, cláusula "limpar_terminal" é uma regra que executa dois comandos shell: "clear" (para sistemas Unix/Linux) e "cls" (para sistemas Windows), que limpam a tela do terminal, a cláusula ":- initialization(menu)" é uma diretiva que define o que deve ser executado quando o programa Prolog é iniciado.

```
% fatos
calcado(007, 'sandalia havaianas slim', sandalia, 35, amarelo, havaianas, 29.99).
calcado(008, 'sapato oxford', sapato, 38, vermelho, vizzano, 149.99).
calcado(009, 'tenis new balance', tenis, 41, preto, 'new balance', 199.99).
calcado(010, 'tenis puma', tenis, 39, azul, puma, 249.99).
calcado(011, 'bota coturno', bota, 37, preto, dakota, 199.99).
calcado(012, 'sandalia salto alto', sandalia, 36, dourado, vizzano, 129.99).
calcado(013, 'sapatilha bico fino', sapatilha, 38, vermelho, moleca, 79.99).
calcado(014, 'tenis asics gel-nimbus', tenis, 40, azul, asics, 399.99).
calcado(015, 'chinelo rider', chinelo, 43, preto, rider, 45.99).
calcado(016, 'sandalia rasteira', sandalia, 39, rosa, moleca, 59.99).
calcado(017, 'sapato salto alto', sapato, 37, preto, vizzano, 159.99).
calcado(018, 'tenis fila disruptor', tenis, 38, branco, fila, 399.99).
calcado(019, 'bota montaria', bota, 40, marrom, bottero, 299.99).
calcado(020, 'sandalia anabela', sandalia, 36, azul, moleca, 99.99).

% imprime calcado
imprime_calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco) :-
    write('Código: '), write(Codigo), nl,
    write('Nome: '), write(Nome), nl,
    write('Tipo: '), write(Tipo), nl,
    write('Tamanho: '), write(Tamanho), nl,
    write('Cor: '), write(Cor), nl,
    write('Marca: '), write(Marca), nl,
    write('Preço: '), write(Preco), nl, nl.
```

```
% menu
menu :-
    nl, write('------ MENU ------'), nl,
    write('1. Mostrar calcado pelo codigo'), nl,
    write('2. Mostrar calçados por tipo e tamanho'), nl,
    write('3. Mostrar calçados por tipo e cor'), nl,
    write('4. Mostrar calçados por tamanho e cor'), nl,
    write('5. Sair'), nl,
    write('-------------------'), nl,
    read(Opcao),
    limpar_terminal,
    escolher_opcao(Opcao).

% mostrar calcado pelo codigo
mostrar_calcado_codigo(Codigo):-
    calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),
    imprime_calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),!.
mostrar_calcado_codigo(_):-
    write('Nao ha calcados com esse codigo'), nl.

% mostrar calçados por tipo e tamanho
mostrar_calcado_tipo_tamanho(Tipo, Tamanho):-
    calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),
    imprime_calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),
    fail.
mostrar_calcado_tipo_tamanho(Tipo, Tamanho) :-
    calcado(_, _, Tipo, Tamanho, _, _, _), !.
mostrar_calcado_tipo_tamanho(_, _):-
    write('Nao ha calcados com esse tipo e tamanho'), nl.
```

```
% mostrar calçados por tipo e cor
mostrar_calcado_tipo_cor(Tipo, Cor):-
    calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),
    imprime_calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),
    fail.
mostrar_calcado_tipo_cor(Tipo, Cor) :-
    calcado(_, _, Tipo, _, Cor, _, _), !.
mostrar_calcado_tipo_cor(_, _):-
    write('Nao ha calcados com esse tipo e cor'), nl.
```

```
% mostrar calçados por tamanho e cor
mostrar_calcado_tamanho_cor(Tamanho, Cor):-
    calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),
    imprime_calcado(Codigo, Nome, Tipo, Tamanho, Cor, Marca, Preco),
    fail.
mostrar_calcado_tamanho_cor(Tamanho, Cor) :-
    calcado(_, _, _, Tamanho, Cor, _, _), !.
mostrar_calcado_tamanho_cor(_, _):-
    write('Nao ha calcados com esse tamanho e cor'), nl.
```

```
% escolher opção
escolher_opcao(1) :-
    write('Digite o codigo do calcado: '), nl,
    read(Codigo), nl,
    mostrar_calcado_codigo(Codigo),
    menu,!.
escolher_opcao(2) :-
    write('Digite o tipo de calcado: '), nl,
    read(Tipo), nl,
    write('Digite o tamanho do calcado: '), nl,
    read(Tamanho), nl,
    mostrar_calcado_tipo_tamanho(Tipo, Tamanho),
    menu,!.
escolher_opcao(3) :-
    write('Digite o tipo de calcado: '), nl,
    read(Tipo), nl,
    write('Digite a cor do calcado: '), nl,
    read(Cor), nl,
    mostrar_calcado_tipo_cor(Tipo, Cor),
    menu,!.
escolher_opcao(4) :-
    write('Digite o tamanho do calcado: '), nl,
    read(Tamanho), nl,
    write('Digite a cor do calcado: '), nl,
    read(Cor), nl,
    mostrar_calcado_tamanho_cor(Tamanho, Cor),
    menu,!.
escolher_opcao(5) :-
    write('Saindo...'), nl.
escolher_opcao(_):-
    nl, write('Opcao invalida'), nl,
    menu
```

```
escolher_opcao(3) :-
    write('Digite o tipo de calcado: '), nl,
    read(Tipo), nl,
    write('Digite a cor do calcado: '), nl,
    read(Cor), nl,
    mostrar_calcado_tipo_cor(Tipo, Cor),
    menu,!.
escolher_opcao(4) :-
    write('Digite o tamanho do calcado: '), nl,
    read(Tamanho), nl,
    write('Digite a cor do calcado: '), nl,
    read(Cor), nl,
    mostrar_calcado_tamanho_cor(Tamanho, Cor),
    menu,!.
escolher_opcao(5) :-
    write('Saindo...'), nl.
escolher_opcao(_):-
    nl, write('Opcao invalida'), nl,
    menu.

limpar_terminal:-
    (shell('clear') ; shell('cls')).

:- initialization(menu).
```

Figura 2. Predicado da primeira questão

2.2. Testes da 1º questão

Nesta seção, serão descritos os experimentos realizados com a linguagem Prolog para a solução de problemas de casos. Em cada teste, será disponibilizado o menu específico responsável pelo experimento.

```
% menu
menu :-
    nl, write('------ MENU ------'), nl,
    write('1. Mostrar calcado pelo codigo'), nl,
```

```
Digite o codigo do calcado:
|: 007.

Código: 7
Nome: sandalia havaianas slim
Tipo: sandalia
Tamanho: 35
Cor: amarelo
Marca: havaianas
Preço: 29.99
```

```
Digite o codigo do calcado:
|: 008.

Código: 8
Nome: sapato oxford
Tipo: sapato
Tamanho: 38
Cor: vermelho
Marca: vizzano
Preço: 149.99
```

```
Digite o codigo do calcado:
|: 009.

Código: 9
Nome: tenis new balance
Tipo: tenis
Tamanho: 41
Cor: preto
Marca: new balance
Preço: 199.99
```

```
Digite o codigo do calcado:
|: 010.

Código: 10
Nome: tenis puma
Tipo: tenis
Tamanho: 39
Cor: azul
Marca: puma
Preço: 249.99
```

```
31      write('2. Mostrar calçados por tipo e tamanho'), nl,
```

```
Digite o tipo de calcado:
|: sandalia.

Digite o tamanho do calcado:
|: 35.

Código: 7
Nome: sandalia havaianas slim
Tipo: sandalia
Tamanho: 35
Cor: amarelo
Marca: havaianas
Preço: 29.99
```

```
Digite o tipo de calcado:
|: sandalia.

Digite o tamanho do calcado:
|: 36.

Código: 12
Nome: sandalia salto alto
Tipo: sandalia
Tamanho: 36
Cor: dourado
Marca: vizzano
Preço: 129.99

Código: 20
Nome: sandalia anabela
Tipo: sandalia
Tamanho: 36
Cor: azul
Marca: moleca
Preço: 99.99
```

```
Digite o tipo de calcado:
|: bota.

Digite o tamanho do calcado:
|: 37.

Código: 11
Nome: bota coturno
Tipo: bota
Tamanho: 37
Cor: preto
Marca: dakota
Preço: 199.99
```

```
Digite o tipo de calcado:
|: bota.

Digite o tamanho do calcado:
|: 38.

Nao ha calcados com esse tipo e tamanho
```

```
Digite o tipo de calcado:
|: sandalia.

Digite o tamanho do calcado:
|: 42.

Nao ha calcados com esse tipo e tamanho

------ MENU ------
1. Mostrar calcado pelo codigo
2. Mostrar calçados por tipo e tamanho
3. Mostrar calçados por tipo e cor
4. Mostrar calçados por tamanho e cor
5. Sair
```

```
32      write('3. Mostrar calçados por tipo e cor'), nl,
```

```
Digite o tipo de calcado:
|: sapatilha.

Digite a cor do calcado:
|: vermelho.

Código: 13
Nome: sapatilha bico fino
Tipo: sapatilha
Tamanho: 38
Cor: vermelho
Marca: moleca
Preço: 79.99
```

```
Digite o tipo de calcado:
|: chinelo.

Digite a cor do calcado:
|: preto.

Código: 15
Nome: chinelo rider
Tipo: chinelo
Tamanho: 43
Cor: preto
Marca: rider
Preço: 45.99
```

```
33      write('4. Mostrar calçados por tamanho e cor'), nl,
```

```
Digite o tamanho do calcado:
|: 38.

Digite a cor do calcado:
|: vermelho.

Código: 8
Nome: sapato oxford
Tipo: sapato
Tamanho: 38
Cor: vermelho
Marca: vizzano
Preço: 149.99

Código: 13
Nome: sapatilha bico fino
Tipo: sapatilha
Tamanho: 38
Cor: vermelho
Marca: moleca
Preço: 79.99
```

```
Digite o tamanho do calcado:
|: 40.

Digite a cor do calcado:
|: marrom.

Código: 19
Nome: bota montaria
Tipo: bota
Tamanho: 40
Cor: marrom
Marca: bottero
Preço: 299.99
```

```
Digite o tamanho do calcado:
|: 42.

Digite a cor do calcado:
|: azul.

Nao ha calcados com esse tamanho e cor
```

```
34      write('5. Sair'), nl,
```

```
Saindo...
true.

?-
```

2.3. Realização da 2º questão

A Figura 3 mostra a contextualização da questão, e, por conseguinte, o que a questão solicita para com a solução, mostrando dados e informações a respeito do problema em questão.

2) Crie uma base de dados fixa chamada FAMÍLIA, o qual deve seer composto por fatos contendo três parâmetros o nome do pais e de um filho, caso os pais tenham mais de um filho, crie um fato para cada filho. E então faça predicados para responder as seguintes questões:
   I. Quais os nomes dos filhos? Imprima uma mensagem caso não tenha filhos.
   II. Quais os nomes dos netos? Imprima uma mensagem caso não tenha netos.
   III. Quais os nomes dos irmãos? Imprima uma mensagem caso não tenha irmãos.
   IV. Quais os nomes dos avós relacionados ao primeiro parâmetro de uma pessoa? E do segundo parâmetro? Imprima uma mensagem caso não tenha avós cadastrados.
   V. Quais os nomes dos tios relacionados ao primeiro parâmetro? E do segundo parâmetro? Imprima uma mensagem caso não tenha tios.

Figura 3. Segunda questão

O código em questão é um programa em Prolog que realiza consultas em uma base de dados de famílias. A base de dados é composta por fatos do tipo "familia(Pessoa1, Pessoa2, Pessoa3)", que significam que Pessoa1 e Pessoa2 são os pais de Pessoa3.

O programa possui cinco predicados: "filhos", "netos", "irmaos", "avos" e "tios". Cada um desses predicados recebe um argumento Pessoa e realiza uma consulta na base de dados para retornar uma informação específica sobre os parentes de Pessoa.

As linhas de 12 – 23, o predicado "filhos" recebe um argumento Pessoa e retorna os nomes dos filhos de Pessoa. Ele realiza duas consultas na base de dados: uma para encontrar os filhos em que Pessoa é pai/mãe e outra para encontrar os filhos em que Pessoa é filho(a). Em seguida, ele escreve o nome de cada filho na tela, seguido de uma quebra de linha. Se Pessoa não tiver filhos na base de dados, o predicado retorna uma mensagem informando isso.

As linhas de 26 – 39, o predicado "netos" recebe um argumento Pessoa e retorna os nomes dos netos de Pessoa. Ele realiza duas consultas na base de dados: uma para encontrar os filhos em que Pessoa é pai/mãe e outra para encontrar os netos dos filhos encontrados na primeira consulta. Em seguida, ele escreve o nome de cada neto

na tela, seguido de uma quebra de linha. Se Pessoa não tiver netos na base de dados, o predicado retorna uma mensagem informando isso.

As linhas de 41 – 70, o predicado "irmaos" recebe um argumento Pessoa e retorna os nomes dos irmãos de Pessoa. Ele realiza várias consultas na base de dados para encontrar irmãos de Pessoa: primeiro, ele encontra os pais de Pessoa e, em seguida, busca por outros filhos desses pais que não sejam Pessoa. Ele escreve o nome de cada irmão na tela, seguido de uma quebra de linha. Se Pessoa não tiver irmãos na base de dados, o predicado retorna uma mensagem informando isso.

As linhas de 71 – 92 O predicado "avos" recebe um argumento Pessoa e retorna os nomes dos avós de Pessoa. Ele realiza várias consultas na base de dados para encontrar os avós maternos e paternos de Pessoa: primeiro, ele encontra os pais de Pessoa, depois encontra os pais desses pais que são avós de Pessoa. Em seguida, ele escreve na tela os nomes dos avós maternos e paternos, separados por uma linha em branco. Se Pessoa não tiver avós na base de dados, o predicado retorna uma mensagem informando isso.

As linhas de 93 – 120, o predicado "tios" recebe um argumento Pessoa e retorna os nomes dos tios de Pessoa. Ele realiza duas consultas na base de dados para encontrar os tios maternos e paternos de Pessoa: primeiro, ele encontra a mãe/pai de Pessoa e, em seguida, busca pelos irmãos desse pai/mãe que não sejam o próprio pai/mãe de Pessoa. Ele escreve o nome de cada tio na tela, seguido de uma quebra de linha. Se Pessoa não tiver tios na base de dados, o predicado retorna uma mensagem informando isso.

```
1  familia(lourenca, romao, zuleide).
2  familia(lourenca, romao, francisca).
3  familia(lourenca, romao, valdira).
4  familia(cecilia, joao, givaldo).
5  familia(cecilia, joao, zemeira).
6  familia(valdira, chicao, fabricio).
7  familia(zuleide, givaldo, vinicius).
8  familia(zuleide, rangel, bruno).
9  familia(leila, givaldo, ana).
10
11
12 % nomes dos filhos
13 filhos(Pessoa) :-
14     (familia(Pessoa, _, Filho);familia(_, Pessoa, Filho)),
15     write(Filho), nl,
16     fail.
17 filhos(Pessoa) :-
18     (familia(Pessoa, _, _);familia(_, Pessoa, _)), nl,!.
19 filhos(Pessoa):-
20     familia(_, _, Pessoa),
21     write(Pessoa), write(' nao possui filhos'), nl,!.
22 filhos(Pessoa):-
23     write(Pessoa), write(' nao está na base'), nl.
```

```
26 % nome dos netos passando uma pessoa no parametro
27 netos(Pessoa):-
28     (familia(Pessoa, _, Filho);familia(_, Pessoa, Filho)),
29     (familia(Filho, _, Neto);familia(_, Filho, Neto)),
30     write(Neto), nl,
31     fail.
32 netos(Pessoa):-
33     (familia(Pessoa, _, Filho);familia(_, Pessoa, Filho)),
34     (familia(Filho, _, _);familia(_, Filho, _)),!.
35 netos(Pessoa):-
36     (familia(Pessoa, _, _);familia(_, Pessoa, _);familia(_, _, Pessoa)),
37     write(Pessoa), write(' nao possui netos'), nl,!.
38 netos(Pessoa):-
39     write(Pessoa), write(' nao esta na base'), nl.
```

```
% nome dos irmaos
irmaos(Pessoa):-
    familia(Mae, Pai, Pessoa),
    familia(Mae, Pai, Irmao),
    Pessoa \= Irmao,
    write(Irmao), nl,
    fail.
irmaos(Pessoa):-
    familia(Mae, Pai, Pessoa),
    familia(Mae, Padrasto, Irmao),
    Pessoa \= Irmao,
    Pai \= Padrasto,
    write(Irmao), nl,
    fail.
irmaos(Pessoa):-
    familia(Mae, Pai, Pessoa),
    familia(Madrasta, Pai, Irmao),
    Pessoa \= Irmao,
    Mae \= Madrasta,
    write(Irmao), nl,
    fail.
irmaos(Pessoa):-
    familia(Mae, Pai, Pessoa),
    (familia(Mae, _, Irmao);familia(_, Pai, Irmao)),
    Pessoa \= Irmao,!.
irmaos(Pessoa):-
    familia(_, _, Pessoa),
    write(Pessoa), write(' nao possui irmaos'), nl,!.
irmaos(Pessoa):-
    write(Pessoa), write(' nao esta na base'), nl.
```

```
% nome dos avos
avos(Pessoa):-
    familia(Mae, Pai, Pessoa),
    familia(GrandMotherMae, GrandFatherMae, Mae),
    familia(GrandMotherPai, GrandFatherPai, Pai),
    write('Avos maternos'), nl, write(GrandMotherMae), nl, write(GrandFatherMae), nl, nl,
    write('Avos paternos'), nl, write(GrandMotherPai), nl, write(GrandFatherPai), nl, nl,!.
avos(Pessoa):-
    familia(Mae, Pai, Pessoa),
    (familia(GrandMotherMae, GrandFatherMae, Mae),
    write('Avos maternos'), nl, write(GrandMotherMae), nl, write(GrandFatherMae), nl, nl,
    write('Avos paternos'), nl, write('Nao possui'), nl, nl;
    familia(GrandMotherPai, GrandFatherPai, Pai),
    write('Avos paternos'), nl, write(GrandMotherPai), nl, write(GrandFatherPai), nl, nl,
    write('Avos maternos'), nl, write('Nao possui'), nl, nl),!.
avos(Pessoa):-
    familia(_, _, Pessoa),
    write(Pessoa), write(' nao possui avos cadastrados na base'), nl,!.
avos(Pessoa):-
    write(Pessoa), write(' nao esta na base'), nl.
```

```
% nome dos tios
tios(Pessoa):-
    familia(Mae, _, Pessoa),
    familia(GrandMotherMae, GrandFatherMae, Mae),
    familia(GrandMotherMae, GrandFatherMae, TioMaterno),
    TioMaterno \= Mae,
    write(TioMaterno), nl,
    fail.
tios(Pessoa):-
    familia(_, Pai, Pessoa),
    familia(GrandMotherPai, GrandFatherPai, Pai),
    familia(GrandMotherPai, GrandFatherPai, TioPaterno),
    TioPaterno \= Pai,
    write(TioPaterno), nl,
    fail.
tios(Pessoa):-
    familia(Mae, Pai, Pessoa),
    (familia(GrandMotherMae, GrandFatherMae, Mae),
    familia(GrandMotherMae, GrandFatherMae, TioMaterno),
    TioMaterno \= Mae;
    familia(GrandMotherPai, GrandFatherPai, Pai),
    familia(GrandMotherPai, GrandFatherPai, TioPaterno),
    TioPaterno \= Pai),!.
tios(Pessoa):-
    familia(_, _, Pessoa),
    write(Pessoa), write(' nao possui tios cadastrados na base'), nl,!.
tios(Pessoa):-
    write(Pessoa), write(' nao esta na base'), nl.
```

```
% menu
menu :-
    nl, write('------ MENU ------'), nl,
    write('1. Mostrar filhos'), nl,
    write('2. Mostrar netos'), nl,
    write('3. Mostrar irmaos'), nl,
    write('4. Mostrar avos maternos e paternos'), nl,
    write('5. Mostrar tios maternos e paternos'), nl,
    write('6. Finalizar programa'), nl,
    write('--------------------'), nl,
    read(Opcao),
    limpar_terminal,
    escolher_opcao(Opcao).
```

```
% escolher opção
escolher_opcao(1) :-
    write('Digite o nome da Pessoa: '), nl,
    read(Pessoa), nl,
    filhos(Pessoa),
    menu,!.
escolher_opcao(2) :-
    write('Digite o nome da Pessoa: '), nl,
    read(Pessoa), nl,
    netos(Pessoa),
    menu,!.
escolher_opcao(3) :-
    write('Digite o nome da Pessoa: '), nl,
    read(Pessoa), nl,
    irmaos(Pessoa),
    menu,!.
escolher_opcao(4) :-
    write('Digite o nome da Pessoa: '), nl,
    read(Pessoa), nl,
    avos(Pessoa),
    menu,!.
escolher_opcao(5) :-
    write('Digite o nome da Pessoa: '), nl,
    read(Pessoa), nl,
    tios(Pessoa),
    menu,!.
escolher_opcao(6):-
    write('Programa finalizado'), nl,!.
escolher_opcao(_) :-
    write('Opcao invalida'), nl,
    menu.
```

```
limpar_terminal:-
    (shell('clear') ; shell('cls')).


:- initialization(menu).
```

Figura 4. Predicado da 2º questão

2.4. Testes da 2º questão

Nesta seção, serão descritos os experimentos realizados com a linguagem Prolog para a solução de problemas de casos. Em cada teste, será disponibilizado o menu específico responsável pelo experimento.

```
% menu
menu :-
    nl, write('------ MENU ------'), nl,
    write('1. Mostrar filhos'), nl,
```

```
Digite o nome da Pessoa:
|: romao.

zuleide
francisca
valdira
```

```
Digite o nome da Pessoa:
|: lourenca.

zuleide
francisca
valdira
```

```
Digite o nome da Pessoa:
|: cecilia.

givaldo
zemeira
```

```
126       write('2. Mostrar netos'), nl,
```

```
Digite o nome da Pessoa:
|: lourenca.

vinicius
bruno
fabricio
```

```
127       write('3. Mostrar irmaos'), nl,
```

```
Digite o nome da Pessoa:
|: zuleide.

francisca
valdira
```

```
Digite o nome da Pessoa:
|: francisca.

zuleide
valdira
```

```
128       write('4. Mostrar avos maternos e paternos'), nl,
```

```
Digite o nome da Pessoa:
|: bruno.

Avos maternos
lourenca
romao

Avos paternos
Nao possui
```

```
Digite o nome da Pessoa:
|: fabricio.

Avos maternos
lourenca
romao

Avos paternos
Nao possui
```

```
129      write('5. Mostrar tios maternos e paternos'), nl,
```

```
Digite o nome da Pessoa:
|: bruno.

francisca
valdira
```

```
Programa finalizado
true.

?-
```

2.5. Realização da 3º questão

A Figura 3 mostra a contextualização da questão, e, por conseguinte, o que a questão solicita para com a solução, mostrando dados e informações a respeito do problema em questão.

3) Crie as seguintes bases de dados dinâmicas com informações geográficas:
   oceano(X)
   pais(X)
   continente(X)
   fronteira(X,Y), onde X e Y podem ser países ou oceanos.
   loc(X,Y), onde X está localizado em Y (X é um país e Y um continente)

   O programa deve conter um menu com as seguintes opções:
   I. Cadastrar oceano, não permitir cadastro repetido;
   II. Cadastrar país, não permitir cadastro repetido;
   III. Cadastrar continente, não permitir cadastro repetido;
   IV. Cadastrar fronteiras, não permitir cadastro se os países ou oceanos não estiverem cadastrados em suas respectivas bases, quando não estiver cadastrado na base perguntar ao usuário se deseja cadastrar;
   V. Cadastrar localização, não permitir cadastro se país e/ou oceano não estiverem cadastrados nas suas respectivas bases, quando não estiver cadastrado na base perguntar ao usuário se deseja cadastrar;
   VI. Localização, dado um país mostre o continente que ele se encontra;
   VII. Países de um Continente, dado um continente mostrar todos os países cadastrados para este continente;
   VIII. Fronteira, onde o usuário digita um país ou um oceano e então mostre todos os países e oceanos em que o mesmo faz fronteira, caso não tenha emitir uma mensagem;
   IX. Mostre todos os pares de países que tem a mesma fronteira;
   X. Mostre todos os países que fazem fronteiras com oceanos;
   XI. Mostre as fronteiras de um determinado oceano.

Figura 3. Cláusulas da primeira questão

A Figura 5, representado por recortes de imagens, formam um predicadoque é um programa em Prolog que contém um menu com várias opções para cadastro e consulta de informações geográficas, como oceanos, países, continentes, fronteiras e localizações.

As linhas de 09 – 24, é um código Prolog que define um menu com diferentes opções para um programa. As opções incluem cadastrar oceanos, países, continentes, fronteiras e localizações, bem como realizar operações.

As linhas de 26 – 35, esse código em Prolog permite que o usuário cadastre vários oceanos até que ele digite 0 para finalizar a inserção. O código solicita que o usuário digite o nome do oceano e verifica se o valor digitado não é 0 e se o oceano ainda não foi cadastrado anteriormente. Se as condições forem atendidas, o oceano é cadastrado e a função cadastrar_oceano é chamada novamente para permitir que o usuário continue a inserir oceanos. Se o usuário digitar 0 ou o oceano já existir, o cadastro de oceanos é encerrado.

As linhas de 37– 46, esse código em Prolog permite que o usuário cadastre vários países até que ele digite 0 para finalizar a inserção. O código solicita que o usuário digite o nome do país e verifica se o valor digitado não é 0 e se o país ainda não foi cadastrado anteriormente. Se as condições forem atendidas, o país é cadastrado e a função cadastrar_pais é chamada novamente para permitir que o usuário continue a inserir países. Se o usuário digitar 0 ou o país já existir, o cadastro de países é encerrado.

As linhas de 48 – 57, esse código em Prolog permite que o usuário cadastre vários continentes até que ele digite 0 para finalizar a inserção. O código solicita que o usuário digite o nome do continente e verifica se o valor digitado não é 0 e se o continente ainda não foi cadastrado anteriormente. Se as condições forem atendidas, o continente é cadastrado e a função cadastrar_continente é chamada novamente para permitir que o usuário continue a inserir continentes. Se o usuário digitar 0 ou o continente já existir, o cadastro de continentes é encerrado.

As linhas de 59 – 77, o código permite que o usuário cadastre várias fronteiras entre países ou oceanos até que ele digite 0 para finalizar a inserção. O código solicita que o usuário digite o nome do primeiro país ou oceano e verifica se o valor digitado não é 0 e se o país ou oceano existe. Em seguida, solicita que o usuário digite o nome do segundo país ou oceano e faz as mesmas verificações. Então, verifica se a fronteira entre esses dois países ou oceanos ainda não foi cadastrada e, se não foi, cadastra a fronteira usando a cláusula assertz. Por fim, exibe uma mensagem de sucesso e chama a si mesma recursivamente para cadastrar novas fronteiras, ou finaliza a inserção caso o usuário digite 0.

As linhas 79 – 97, é responsável por cadastrar várias localizações (um país e um continente) até que o usuário digite "0" para finalizar a inserção. Antes de cadastrar a localização, ele verifica se o país e o continente já foram cadastrados anteriormente e se a localização ainda não existe. Se tudo estiver ok, ele insere a localização na base de dados. Quando o usuário digita "0", a inserção é finalizada.

As linhas 99 – 104, esta regra recebe como entrada o nome de um país e verifica se este está cadastrado na base de dados. Se estiver, a regra busca a informação de qual continente ele está localizado e exibe a informação na tela. Caso contrário, a regra informa que não há informação de localização para o país informado.

As linhas 106 – 114, essa cláusula tem como objetivo mostrar na tela todos os países que estão localizados em um determinado continente.

As linhas 117 – 125, essa é uma cláusula que tem como objetivo mostrar todas as fronteiras de um país ou oceano e utiliza-se do fail.

As linhas 127 – 137, esse código tem o caso de mostrar todos os países que fazem fronteira com oceanos. Primeiro, o predicado procura fronteiras entre países e oceanos, verificando se o segundo elemento é um oceano usando o predicado "oceano". Depois, ele escreve na tela os países e oceanos encontrados e falha, para continuar a busca.

As linhas 139 – 150, O código acima tem como objetivo mostrar todos os países que fazem fronteira com dois países especificados. Possui três cláusulas:

- A primeira cláusula é executada quando há fronteira entre os países Pais1 e Pais2. Nesse caso, a regra falha (fail) para que sejam exibidos todos os países que fazem fronteira com ambos Pais1 e Pais2.
- A segunda cláusula é executada quando há fronteira entre os países Pais1 e Pais2, mas nenhum país em comum com esses dois países. Nesse caso, a regra é bem-sucedida (true) e nenhum país é exibido.
- A terceira cláusula é executada quando não há fronteira entre os países Pais1 e Pais2. Nesse caso, uma mensagem é exibida informando que não há fronteiras comuns entre esses países.

As linhas 216 – 227, essa é uma função que verifica se um determinado país está presente na base de dados do programa. Se o país estiver cadastrado na base, a função escreve uma mensagem informando que o país foi encontrado. Caso contrário, a função pergunta ao usuário se deseja cadastrar o país.

As linhas 229 – 240, esse predicado verifica se um oceano existe na base de dados. Se existir, imprime uma mensagem informando que o oceano foi encontrado. Caso contrário, imprime uma mensagem informando que o oceano ainda não foi cadastrado e pergunta se deseja cadastrar. Se a resposta for sim, adiciona o oceano à base de dados usando o predicado assertz(oceano(Oceano)).

As linhas 242 – 254, essa cláusula é um predicado que verifica se um dado continente existe na base de dados. Ele possui duas cláusulas.

1. A primeira cláusula é executada se o continente já estiver cadastrado na base de dados. Ela verifica se o continente fornecido como entrada está presente na base de dados, e caso esteja, escreve uma mensagem informando que o continente foi encontrado na base de dados.
2. A segunda cláusula é executada caso o continente ainda não esteja cadastrado na base de dados. Ela exibe uma mensagem informando que o continente não está cadastrado.

As linhas 256 – 259, essa cláusula tem o objetivo de verificar se uma relação de fronteira entre duas entidades está presente na base de dados.

As linhas 264 – 278, esta cláusula tem a função de verificar se uma string passada como argumento representa um oceano ou um país, verificando se existe na base de dados. Caso exista, a cláusula retorna verdadeiro. Caso contrário, ela pergunta ao usuário se deseja cadastrar a entrada. Se o usuário desejar cadastrar, a cláusula pergunta se a entrada é um oceano ou um país e a cadastra.

As linhas 280 – 283, essa cláusula recebe um país e um continente como argumentos e verifica se eles já estão na base de dados da localização.

```
:- dynamic(oceano/1).
:- dynamic(pais/1).
:- dynamic(continente/1).
:- dynamic(fronteira/2).
:- dynamic(localizacao/2).

% menu
menu :-
    nl, write('------ MENU ------'), nl,
    write('1. Cadastrar oceano'), nl,
    write('2. Cadastrar país'), nl,
    write('3. Cadastrar continente'), nl,
    write('4. Cadastrar fronteiras'), nl,
    write('5. Cadastrar localizacao'), nl,
    write('6. Localizacao'), nl,
    write('7. Paises de um continente'), nl,
    write('8. Fronteira'), nl,
    write('9. Pares de paises que tem a mesma fronteira'), nl,
    write('10. Mostrar todos os paises que fazem fronteira com oceanos'), nl,
    write('11. fronteiras de um oceano'), nl,
    write('0. Encerrar programa'), nl,
    write('--------------------'), nl,
    read(Opcao), limpar_terminal,
    escolher_opcao(Opcao).
```

```
% cadastrar varios oceanos ate o usuario digitar 0 para finalizar a insercao
cadastrar_oceano:-
    write('Digite 0 caso queira finalizar a insercao de oceanos'), nl,
    write('Digite o nome do oceano: '), nl,
    read(Oceano),
    Oceano \= 0,
    oceanoExiste(Oceano),
    cadastrar_oceano,!.
cadastrar_oceano:-
    nl, write('Finalizando insercao de oceanos'), nl, nl.
```

```
% cadastrar varios paises ate o usuario digitar 0 para finalizar a insercao
cadastrar_pais:-
    write('Digite 0 caso queira finalizar a insercao de paises'), nl,
    write('Digite o nome do pais: '), nl,
    read(Pais),
    Pais \= 0,
    paisExiste(Pais),
    cadastrar_pais,!.
cadastrar_pais:-
    nl, write('Finalizando insercao de paises'), nl, nl.
```

```
% cadastrar varias fronteiras ate o usuario digitar 0 para finalizar a insercao
% cadastrar varias localizacoes ate o usuario digitar 0 para finalizar a insercao
% a localizacao deve ser entre um pais e um continente que ja foram cadastrados
% buscar se o pais e o continente existem antes de cadastrar a localizacao
cadastrar_localizacao:-
    write('Digite 0 caso queira finalizar a insercao de localizacoes'), nl,
    write('Digite o nome do pais ou oceano: '), nl,
    read(Info1),
    Info1 \= 0,
    oceanoPais(Info1),
    write('Digite o nome do continente: '), nl,
    read(Continente),
    Continente \= 0,
    continenteExiste(Continente),
    \+ localizacaoExiste(Info1,Continente),
    assertz(localizacao(Info1,Continente)),
    write('Localizacao cadastrada com sucesso'), nl, nl,
    cadastrar_localizacao,!.
cadastrar_localizacao:-
    nl, write('Finalizando insercao de localizacoes'), nl, nl.
```

```
% mostrar localizacao de um pais
localizacaoPais(Pais):-
    localizacao(Pais,Continente),
    nl, write(Pais), write(' esta localizado no continente '), write(Continente), nl,!.
localizacaoPais(Pais):-
    nl, write(Pais), write(' nao tem localizacao definida na base'), nl.
```

```
% mostrar paises de um continente
paisesContinente(Continente):-
    localizacao(Pais,Continente),
    write(Pais), nl,
    fail.
paisesContinente(Continente):-
    localizacao(_,Continente),!.
paisesContinente(Continente):-
    nl, write(Continente), write(' nao possui paises cadastrados na base'), nl.
```

```
% mostrar todas as fronteiras de um pais ou oceano
fronteirasPaisesOceanos(Info):-
    (fronteira(Info,Fronteira); fronteira(Fronteira, Info)),
    write(Info), write(' faz fronteira com '), write(Fronteira), nl,
    fail.
fronteirasPaisesOceanos(Info):-
    (fronteira(Info,_); fronteira(_, Info)),!.
fronteirasPaisesOceanos(Info):-
    nl, write(Info), write(' nao possui fronteira cadastrada'), nl.
```

```
% mostrar todos os paises que fazem fronteira com oceanos
fronteiraComOceanos:-
    (fronteira(Pais,Oceano), oceano(Oceano);
    fronteira(Oceano,Pais), oceano(Oceano)),
    write(Pais), write(' faz fronteira com '), write(Oceano), nl,
    fail.
fronteiraComOceanos:-
    (fronteira(_,Oceano), oceano(Oceano);
    fronteira(Oceano,_), oceano(Oceano)),!.
fronteiraComOceanos:-
    nl, write('Nao ha fronteiras entre paises e oceanos cadastradas'), nl.
```

```
% mostrar todos os paises que fazem fronteira com o par de paises informado
fronteirasEmComum(Pais1,Pais2):-
    (fronteira(Pais1,PaisFronteira) ; fronteira(PaisFronteira,Pais1)),
    PaisFronteira \= Pais2,
    (fronteira(Pais2,PaisFronteira) ; fronteira(PaisFronteira,Pais2)),
    write(Pais1), write(' e '), write(Pais2), write(' fazem fronteira com '), write(PaisFronteira), nl, fail.
fronteirasEmComum(Pais1,Pais2):-
    (fronteira(Pais1,PaisFronteira) ; fronteira(PaisFronteira,Pais1)),
    PaisFronteira \= Pais2,
    (fronteira(Pais2,PaisFronteira) ; fronteira(PaisFronteira,Pais2)),!.
fronteirasEmComum(Pais1,Pais2):-
    nl, write(Pais1), write(' e '), write(Pais2), write(' nao fazem fronteira com nenhum pais em comum'), nl.
```

```
% verificar se pais esta na base de dados
paisExiste(Pais):-
    pais(Pais),
    nl, write('Pais encontrado na base'), nl, nl,!.
paisExiste(Pais):-
    nl, write(Pais), write(' nao cadastrado'), nl, nl,
    write('Deseja cadastrar?'), nl,
    write('1 - Sim'), nl,
    write('0 - Nao'), nl,
    read(Opcao),
    Opcao = 1,
    assertz(pais(Pais)).
```

```
% verificar se oceano esta na base de dados
oceanoExiste(Oceano):-
    oceano(Oceano),
    nl, write('Oceano encontrado na base'), nl, nl,!.
oceanoExiste(Oceano):-
    nl, write(Oceano), write(' ainda nao cadastrado'), nl, nl,
    write('Deseja cadastrar?'), nl,
    write('1 - Sim'), nl,
    write('0 - Nao'), nl,
    read(Opcao),
    Opcao = 1,
    assertz(oceano(Oceano)).
```

```
% verificar se continente esta na base de dados
% se nao estiver, perguntar se o usuario deseja cadastrar
continenteExiste(Continente):-
    continente(Continente),
    nl, write(Continente), write(' encontrado na base'), nl, nl,!.
continenteExiste(Continente):-
    nl, write(Continente), write(' nao cadastrado'), nl, nl,
    write('Deseja cadastrar?'), nl,
    write('1 - Sim'), nl,
    write('0 - Nao'), nl,
    read(Opcao),
    Opcao = 1,
    assertz(continente(Continente)).
```

```
% verificar se fronteira esta na base de dados
fronteiraExiste(Info1,Info2):-
    (fronteira(Info1,Info2) ; fronteira(Info2,Info1)),
    nl, write('Fronteira encontrada na base'), nl, nl.
```

```
% verificar se oceano ou pais foi digitado
oceanoPais(S):-
    (oceano(S); pais(S)),!.
oceanoPais(S):-
    nl, write('Oceano ou pais nao encontrado'), nl, nl,
    write('Deseja cadastrar?'), nl,
    write('1 - Sim'), nl,
    write('0 - Nao'), nl,
    read(Opcao),
    Opcao = 1,
    write('Eh um oceano ou um pais?'), nl,
    write('1 - Oceano'), nl,
    write('2 - Pais'), nl,
    read(Opcao2),
    (Opcao2 = 1, assertz(oceano(S)) ; Opcao2 = 2, assertz(pais(S))).
```

```
280  % verificar se localização esta na base de dados
281  localizacaoExiste(Pais,Continente):-
282      localizacao(Pais,Continente),
283      nl, write('Localizacao encontrada na base'), nl, nl.
```

2.6. Testes da 3º questão

```
7   % menu
8   menu :-
9       nl, write('------ MENU ------'), nl,
10      write('1. Cadastrar oceano'), nl,
```

```
Digite 0 caso queira finalizar a insercao de oceanos
Digite o nome do oceano:
|: atlantico.

atlantico ainda nao cadastrado

Deseja cadastrar?
1 - Sim
0 - Nao
```

```
Digite 0 caso queira finalizar a insercao de oceanos
Digite o nome do oceano:
|: pacifico.

pacifico ainda nao cadastrado

Deseja cadastrar?
1 - Sim
0 - Nao
|: Sim.
```

```
11      write('2. Cadastrar país'), nl,
```

```
Digite 0 caso queira finalizar a insercao de paises
Digite o nome do pais:
|: brasil.

Pais encontrado na base

Digite 0 caso queira finalizar a insercao de paises
Digite o nome do pais:
|: argentina.

argentina nao cadastrado

Deseja cadastrar?
1 - Sim
0 - Nao
|: Sim.
```

```
12      write('3. Cadastrar continente'), nl,
```

```
Digite 0 caso queira finalizar a insercao de continentes
Digite o nome do continente:
|: america_do_sul.

america_do_sul nao cadastrado

Deseja cadastrar?
1 - Sim
0 - Nao
|: Sim.
```

```
Digite 0 caso queira finalizar a insercao de continentes
Digite o nome do continente:
|: america_do_norte.

america_do_norte nao cadastrado

Deseja cadastrar?
1 - Sim
0 - Nao
|: Sim.
```

```
15      write('6. Localizacao'), nl,
```

```
Digite o nome do pais ou oceano:
|: brasil.

brasil esta localizado no continente america_do_sul
```

```
13      write('4. Cadastrar fronteiras'), nl,
```

```
Digite 0 caso queira finalizar a insercao de fronteiras
Digite o nome do pais ou oceano:
|: brasil.
Digite o nome do 2º pais ou oceano:
|: paraguai.

Fronteira cadastrada com sucesso

Digite 0 caso queira finalizar a insercao de fronteiras
Digite o nome do pais ou oceano:
|: brasil.
Digite o nome do 2º pais ou oceano:
|: bolivia.

Fronteira cadastrada com sucesso
```

```
14        write('5. Cadastrar localizacao'), nl,
```

```
Digite 0 caso queira finalizar a insercao de localizacoes
Digite o nome do pais ou oceano:
|: brasil.
Digite o nome do continente:
|: america_do_sul.

america_do_sul encontrado na base

Localizacao cadastrada com sucesso

Digite 0 caso queira finalizar a insercao de localizacoes
Digite o nome do pais ou oceano:
|: argentina.
Digite o nome do continente:
|: america_do_sul.

america_do_sul encontrado na base

Localizacao cadastrada com sucesso

Digite 0 caso queira finalizar a insercao de localizacoes
Digite o nome do pais ou oceano:
|: chile.
Digite o nome do continente:
|: america_do_sul.

america_do_sul encontrado na base

Localizacao cadastrada com sucesso

Digite 0 caso queira finalizar a insercao de localizacoes
Digite o nome do pais ou oceano:
|: peru.
Digite o nome do continente:
|: america_do_sul.

america_do_sul encontrado na base

Localizacao cadastrada com sucesso
```

```
16        write('7. Paises de um continente'), nl,
```

```
Digite o nome do continente:
|: america_do_sul.


america_do_sul encontrado na base

brasil
argentina
chile
peru
```

```
17        write('8. Fronteira'), nl,
```

```
Digite o nome do pais ou oceano:
|: brasil.


Pais encontrado na base

brasil faz fronteira com argentina
brasil faz fronteira com uruguai
brasil faz fronteira com paraguai
brasil faz fronteira com bolivia
brasil faz fronteira com colombia
brasil faz fronteira com peru
brasil faz fronteira com chile
```

```
18      write('9. Pares de paises que tem a mesma fronteira'), nl,
```

```
Digite o nome do primeiro pais:
|: brasil.


Pais encontrado na base

Digite o nome do segundo pais:
|: argentina.


Pais encontrado na base

brasil e argentina fazem fronteira com uruguai
```

```
19      write('10. Mostrar todos os paises que fazem fronteira com oceanos'), nl,
20      write('11. fronteiras de um oceano'), nl,
21      write('0. Encerrar programa'), nl,
22      write('--------------------'), nl,
23      read(Opcao), limpar_terminal,
24      escolher_opcao(Opcao).
```

```
Digite o nome de um oceano
|: atlantico.


Oceano encontrado na base

atlantico faz fronteira com brasil
```

```
brasil faz fronteira com atlantico
chile faz fronteira com pacifico
peru faz fronteira com pacifico
```

```
Finalizando programa
true.

?-
```

3. Conclusão

Consequentemente, pode-se concluir que os algoritmos utilizados neste trabalho foram eficientes na execução dos testes propostos, atingindo assim o objetivo almejado. Além disso, a realização dos experimentos com a ferramenta Swi Prolog permitiu aprender que esta linguagem é declarativa, diferentemente de linguagens procedurais ou orientadas a objetos, que exigem passos específicos para solucionar um problema. Em vez disso, o Prolog fornece uma descrição do problema a ser resolvido por meio de fatos e regras lógicas. Em suma, este estudo demonstrou a importância e a eficácia do uso de linguagens declarativas como o Prolog na resolução de problemas.