
UFPE

# Relatório
# Lógica para Computação (IF972)

Orientador: Dr. Sergio Queiroz

Equipe: Davy de Andrade Mota
Eduardo Santos de Moura
Rafael Rodrigues Ferreira Teles

Sumário

1. Implementação …… 2.
2. Traces ..…….. 7.
3. Testes…….. 10.

# 1. Implementação.

***Passo 1:***

```
lugar(serraDoCipo,[inverno,primavera],diversao,dificil).
lugar(paoDeAcucar,[verao,outono,inverno,primavera],esporte,dificil).
lugar(saoBentoDoSapucai,[inverno,primavera],esporte,dificil).
lugar(pindamonhangaba,[inverno,primavera],esporte,medio).
lugar(chapadaDaDiamantina,[inverno,primavera],diversao,medio).
lugar(lapinha,[inverno,primavera],esporte,medio).
lugar(serraCaiada,[primavera,verao],diversao,facil).
lugar(pedraDaBoca,[primavera,verao],diversao,facil).
lugar(lapaDoSeuAntao,[verao,outono,inverno,primavera],esporte,facil).
```

Neste passo foram implementados fatos correspondentes aos dados dos locais de escalada. Ex: `lugar(Nome,[Estações],Tipo,Dificuldade).`

***Passo 2:***

```
getEstacao(X,Y,outono) :- (X>=20,X=<31,(Y=marco;Y=3));(X>0,X=<30,(Y=abril;Y=4));(X>0,X=<31,(Y=maio;Y=5));(X>0,X<20,(Y=junho;Y=6)).
getEstacao(X,Y,inverno) :- (X>=20,X=<30,(Y=junho;Y=6));(X>0,X=<31,(Y=julho;Y=7));(X>0,X=<31,(Y=agosto;Y=8));(X>0,X<22,(Y=setembro;Y=9)).
getEstacao(X,Y,primavera) :- (X>=22,X=<30,(Y=setembro;Y=9));(X>0,X=<31,(Y=outubro;Y=10));(X>0,X=<30,(Y=novembro;Y=11));(X>0,X<21,(Y=dezembro;Y=12)).
getEstacao(X,Y,verao) :- (X>=21,X=<31,(Y=dezembro;Y=12));(X>0,X=<31,(Y=janeiro;Y=1));(X>0,X=<29,(Y=fevereiro;Y=2));(X>0,X<20,(Y=marco;Y=3)).
```

Foi definida a regra `getEstacao`, que recebe como parâmetros o dia e mês do ano e retorna sua estação correspondente.

Ex:

Entrada: `getEstacao(8,fevereiro,Y).`

Saída: `Y = verao`

***Passo 3*:**

```
:-dynamic(getEscalador/2).
setEscalador(X,Y) :- asserta(getEscalador(X,Y)).
```

A regra `setEscalador` foi criada com a finalidade de cadastrar os escaladores armazenando-os dentro de `getEscalador`, um predicado dinâmico.

Ex:
```
:-dynamic(getEscalador/2).
setEscalador(Nome,Marinheiros):-
asserta(getEscalador(Nome,Marinheiros)).
```

A função `asserta` foi utilizada com o intuito de armazenar o predicado dinâmico `getEscalador`. Também foi usado `:-dynamic(getEscalador/2).` para definir `getEscalator` como um predicado dinâmico.

***Passo 4:***

```
forma(X,fraca) :- \+(getEscalador(X,_)).
forma(X,fraca) :- getEscalador(X,Y),Y>=0,Y<10.
forma(X,media) :- getEscalador(X,Y),Y>=10,Y<25.
forma(X,atletica) :- getEscalador(X,Y),Y>=25.
```

Para ser feita essa verificação de condicionamento físico, foi implementada a regra `forma`, que dado o nome do escalador, retorna a sua forma física no segundo parâmetro. Com base no nome, `forma` irá buscar a quantidade de marinheiros em `getEscalador`, que foi previamente definida. Caso o nome não esteja cadastrado, o `getEscalador` retornará `false`, se encaixando no primeiro caso.

***Passo 5:***

```
getNivel(X,facil) :- X>=0,X<16.
getNivel(X,medio) :- X>=16,X<40.
getNivel(X,dificil) :- X>=40.
```

Regra que define o nível de dificuldade correspondente ao escalador. Recebe como parâmetro o número de horas escaladas (X) e dependendo das restrições o indivíduo deve se encaixar em três categorias: fácil, médio ou difícil.

Ex:

Entrada: `getNivel(1,Y).`

Saída: `Y = fraco`

***Passo 6:***

```
checkDificuldade(X,Y,facil) :- forma(X,Forma),(Forma=fraca;Forma=media;Forma=atletica),getNivel(Y,Nivel),(Nivel=facil;Nivel=medio;Nivel=dificil).
checkDificuldade(X,Y,medio) :- forma(X,Forma),(Forma=media;Forma=atletica),getNivel(Y,Nivel),(Nivel=medio;Nivel=dificil).
checkDificuldade(X,Y,dificil) :- forma(X,Forma),(Forma=atletica),getNivel(Y,Nivel),(Nivel=dificil).
```

Foi implementada a função `checkDificuldade` que recebe como entrada o nome do escalador cadastrado (ou não) e as horas praticadas, retornando o último parâmetro correspondente a dificuldade a qual o escalador se encaixa.

Ex:

Entrada: `setEscalador(sergio, 70).`
`checkDificuldade(sergio,100,Y).`

Saída: `Y = fraco;`
`Y = medio;`
`Y = dificil;`

***Passo 7:***

```
checkTipo(X,Y) :- lugar(Y,_,X,_).
```

`checkTipo` tem como função única buscar nos axiomas previamente estipulados os tipos correspondentes aos seus lugares. Recebe como entrada o tipo de atividade e retorna todos os lugares que possuem o mesmo tipo de atividades dadas.

Ex:

Entrada: `checkTipo(diversao, Z).`

Saída: `Z = serraDoCipo ;`
`Z = chapadaDaDiamantina ;`
`Z = serraCaiada ;`
`Z = pedraDaBoca.`

***Passo 8:***

```
pertence(X,[X|Z]). % Caso Base
pertence(X,[W|Z]) :- pertence(X,Z). % Passo Recursivo
checkEstacao(X,Y) :- lugar(Y,Z,_,_),pertence(X,Z).
```

Passo recursivo que verifica se um elemento está dentro de uma lista. Recebe como entrada a estação (X) e retorna um tipo de dados `booleano`. Primeiramente se considera que o elemento procurado se encontra na cabeça na lista (caso base). Neste caso, ele retornará `True`. Caso o contrário, o programa procederá recursivamente, comparando o elemento procurado com todo o resto da lista original exceto sua cabeça (passo recursivo).

Ex:

Entrada: `pertence(1,[0,1,2,3]).`

Saída: `true`

***Passo 9:***

```
:-dynamic(getDupla/1).
setDupla(X) :- assert(getDupla(X)).
```

Cadastra as duplas (Lugar,Dificuldade) que serão posteriormente geradas pela função `recomendarEscalada()`.

***Passo 10:***

```
recomendarEscalada(Nome,Horas,Tipo,Dia,Mes,Lista) :-
  getEstacao(Dia,Mes,Estacao),
  checkDificuldade(Nome,Horas,DificuldadePessoa),
  checkTipo(Tipo,LugarTipo),
  checkEstacao(Estacao,LugarEstacao),
  LugarTipo = LugarEstacao,
  lugar(LugarTipo,_,_,Dificuldade),
  Dificuldade = DificuldadePessoa,
  Lista = (LugarTipo,Dificuldade),
  setDupla(Lista).
```

Por fim, foi implementado o predicado principal `recomendarEscalada`.Ele utiliza como base todas as implementações anteriores e retorna as opções de locais para escalada. Primeiro recolhe o valor do dia e do mês e guarda a estação na variável `Estacao`. Depois os valores do nome e horas praticadas são coletados e as possibilidades de dificuldades são armazenadas na variável
. Checa-se a os lugares com determinado tipo e determinada estação, e apenas os lugares que coincidem permanecerão. Armazena-se as dificuldades referentes aos respectivos lugares que permaneceram e depois compara-se as dificuldades dos lugares com os níveis de dificuldade que o indivíduo tem acesso. No final, cadastra todas as possibilidades de duplas pelo `setDupla`.

***Passo 11:***

```
getLista(Lista) :- findall(X,getDupla(X),Lista), retractall(getDupla(_)).
```

Junta com o `findall` todas as duplas cadastradas em uma única lista e depois elimina todos os predicados dinâmicos `getDupla` para que múltiplas consultas possam ser realizadas.

## 2. Traces

### getEstacao(Dia,Mês,Estação)

```
[trace] 2 ?- getEstacao(20,abril,Y).
   Call: (7) getEstacao(20, abril, _G1820) ? creep
   Call: (8) 20>=20 ? creep
   Exit: (8) 20>=20 ? creep
   Call: (8) 20=<31 ? creep
   Exit: (8) 20=<31 ? creep
   Call: (8) abril=marco ? creep
   Fail: (8) abril=marco ? creep
   Redo: (7) getEstacao(20, abril, outono) ? creep
   Call: (8) 20>0 ? creep
   Exit: (8) 20>0 ? creep
   Call: (8) 20=<30 ? creep
   Exit: (8) 20=<30 ? creep
   Call: (8) abril=abril ? creep
   Exit: (8) abril=abril ? creep
   Exit: (7) getEstacao(20, abril, outono) ? creep
Y = outono
```

### getNivel(Horas,Dificuldade)

```
[trace] 3 ?- getNivel(42,Y).
   Call: (7) getNivel(42, _G1771) ? creep
   Call: (8) 42>=0 ? creep
   Exit: (8) 42>=0 ? creep
   Call: (8) 42<16 ? creep
   Fail: (8) 42<16 ? creep
   Redo: (7) getNivel(42, _G1771) ? creep
   Call: (8) 42>=16 ? creep
   Exit: (8) 42>=16 ? creep
   Call: (8) 42<40 ? creep
   Fail: (8) 42<40 ? creep
   Redo: (7) getNivel(42, _G1771) ? creep
   Call: (8) 42>=40 ? creep
   Exit: (8) 42>=40 ? creep
   Exit: (7) getNivel(42, dificil) ? creep
Y = dificil.
```

**setEscalador(Nome,Horas)**

```
[trace] 5 ?-
|    setEscalador(sergio,70).
   Call: (7) setEscalador(sergio, 70) ? creep
^  Call: (8) asserta(getEscalador(sergio, 70)) ? creep
^  Exit: (8) asserta(getEscalador(sergio, 70)) ? creep
   Exit: (7) setEscalador(sergio, 70) ? creep
true.
```

**forma(Pessoa,Forma)**

```
[trace] 3 ?- forma(sergio,X).
   Call: (7) forma(sergio, _G1777) ? creep
   Call: (8) getEscalador(sergio, _G1853) ? creep
   Exit: (8) getEscalador(sergio, 70) ? creep
   Call: (8) getEscalador(sergio, _G1853) ? creep
   Exit: (8) getEscalador(sergio, 70) ? creep
   Call: (8) 70>=0 ? creep
   Exit: (8) 70>=0 ? creep
   Call: (8) 70<10 ? creep
   Fail: (8) 70<10 ? creep
   Redo: (7) forma(sergio, _G1777) ? creep
   Call: (8) getEscalador(sergio, _G1853) ? creep
   Exit: (8) getEscalador(sergio, 70) ? creep
   Call: (8) 70>=10 ? creep
   Exit: (8) 70>=10 ? creep
   Call: (8) 70<25 ? creep
   Fail: (8) 70<25 ? creep
   Redo: (7) forma(sergio, _G1777) ? creep
   Call: (8) getEscalador(sergio, _G1853) ? creep
   Exit: (8) getEscalador(sergio, 70) ? creep
   Call: (8) 70>=25 ? creep
   Exit: (8) 70>=25 ? creep
   Exit: (7) forma(sergio, atletica) ? creep
X = atletica.
```

**checkDificuldade(Nome,Horas,Dificuldade)**

```
| 
|     checkDificuldade(sergio,100,Z).
   Call: (7) checkDificuldade(sergio, 100, _G3039) ? creep
   Call: (8) forma(sergio, _G3116) ? creep
   Call: (9) getEscalador(sergio, _G3116) ? creep
   Exit: (9) getEscalador(sergio, 70) ? creep
   Call: (9) getEscalador(sergio, _G3116) ? creep
   Exit: (9) getEscalador(sergio, 70) ? creep
   Call: (9) 70>=0 ? creep
   Exit: (9) 70>=0 ? creep
   Call: (9) 70<10 ? creep
   Fail: (9) 70<10 ? creep
   Redo: (8) forma(sergio, _G3116) ? creep
   Call: (9) getEscalador(sergio, _G3116) ? creep
   Exit: (9) getEscalador(sergio, 70) ? creep
   Call: (9) 70>=10 ? creep
   Exit: (9) 70>=10 ? creep
   Call: (9) 70<25 ? creep
   Fail: (9) 70<25 ? creep
   Redo: (8) forma(sergio, _G3116) ? creep
   Call: (9) getEscalador(sergio, _G3116) ? creep
   Exit: (9) getEscalador(sergio, 70) ? creep
   Call: (9) 70>=25 ? creep
   Exit: (9) 70>=25 ? creep
   Exit: (8) forma(sergio, atletica) ? creep
   Call: (8) atletica=fraca ? creep
   Fail: (8) atletica=fraca ? creep
   Redo: (7) checkDificuldade(sergio, 100, facil) ? creep
   Call: (8) atletica=media ? creep
   Fail: (8) atletica=media ? creep
   Redo: (7) checkDificuldade(sergio, 100, facil) ? creep
   Call: (8) atletica=atletica ? creep
   Exit: (8) atletica=atletica ? creep
   Call: (8) getNivel(100, _G3116) ? creep
   Call: (9) 100>=0 ? creep
   Exit: (9) 100>=0 ? creep
   Call: (9) 100<16 ? creep
   Fail: (9) 100<16 ? creep
   Redo: (8) getNivel(100, _G3116) ? creep
   Call: (9) 100>=16 ? creep
   Exit: (9) 100>=16 ? creep
   Call: (9) 100<40 ? creep
   Fail: (9) 100<40 ? creep
   Redo: (8) getNivel(100, _G3116) ? creep
   Call: (9) 100>=40 ? creep
   Exit: (9) 100>=40 ? creep
   Exit: (8) getNivel(100, dificil) ? creep
   Call: (8) dificil=facil ? creep
   Fail: (8) dificil=facil ? creep
   Redo: (7) checkDificuldade(sergio, 100, facil) ? creep
   Call: (8) dificil=medio ? creep
   Fail: (8) dificil=medio ? creep
   Redo: (7) checkDificuldade(sergio, 100, facil) ? creep
   Call: (8) dificil=dificil ? creep
   Exit: (8) dificil=dificil ? creep
   Exit: (7) checkDificuldade(sergio, 100, facil) ? creep
Z = facil .
```

## 3.Testes

### Caso  com escalador cadastrado:

```
2 ?- setEscalador(sergio,34).
true.

3 ?- recomendarEscalada(sergio,50,diversao,3,outubro,X).
X = (serraCaiada, facil) ;
X = (pedraDaBoca, facil) ;
X = (chapadaDaDiamantina, medio) ;
X = (serraDoCipo, dificil) ;
false.

4 ?- getLista(Lista).
Lista = [ (serraCaiada, facil), (pedraDaBoca, facil), (chapadaDaDiamantina, medio), (serraDoCipo, dificil)].
```

### Caso  com escalador sem cadastro:

```
5 ?- recomendarEscalada(carlos,50,diversao,3,outubro,X).
X = (serraCaiada, facil) ;
X = (pedraDaBoca, facil) ;
false.

6 ?- getLista(Lista).
Lista = [ (serraCaiada, facil), (pedraDaBoca, facil)].
```

**OBS: os meses estão sendo dados em sua forma escrita.**

### Caso adicional com escalador cadastrado:

```
7 ?- setEscalador(joao,23).
true.

8 ?- recomendarEscalada(joao,12,esporte,8,12,X).
X = (lapaDoSeuAntao, facil) ;
false.

9 ?- getLista(Lista).
Lista = [ (lapaDoSeuAntao, facil)].
```

**OBS: os meses estão sendo dados em sua forma numérica.**

### Caso adicional com escalador sem cadastrado:

```
17 ?- recomendarEscalada(renan,55,diversao,22,5,X).
false.

18 ?- getLista(Lista).
Lista = [].
```

**OBS: neste caso, não existe nenhuma opção de escalada para o indivíduo. Neste caso, uma lista vazia é retornada.**