

# Roteiro IV

Alexsandro Santos Soares

prof.asoares@gmail.com

Programação Lógica  
Faculdade de Computação  
Universidade Federal de Uberlândia

27 de março de 2021

Este roteiro tem por finalidade:

- Continuar a prática de programação recursiva.

## 1 Recursividade

**Ex. 1** Defina um predicado `soma_acum(L,K)` que, dado uma lista `L` de inteiros, retorna uma lista `K` na qual cada elemento é a soma de todos os elementos em `L` até a mesma posição. Exemplo:

```
?- soma_acum([1,2,3,4],K).  
K = [1,3,6,10].
```

**Ex. 2** Escreva um predicado `soma_até(N,S)` que calcule a soma de todos os números entre 1 e `N`.

```
?- soma_até(5,S).  
S = 15.
```

**Ex. 3** Defina um predicado `dec_para_bin(N,B)` que converte um número natural `N` em sua representação binária `B`, uma lista. Por exemplo:

```
?- dec_para_bin(51,S).  
S = [1,1,0,0,1,1].
```

**Ex. 4** Escreva um programa para `sem_repeticao(L1,L2)`, com `L2` sendo o resultado da remoção de todos os elementos repetidos de `L1`. Por exemplo,

```
?- sem_repeticao([a,b,c,b], [a,c,b]).  
true
```

Dica: use `member`

**Ex. 5** Projete um predicado `segmento(S,L)` que testa se o seu primeiro argumento é um segmento contínuo contido em qualquer parte da lista `L`. Como exemplo,

```
?- segmento([a,b,c],[1,c,a,b,c,3]).
true

?- segmento([a,b],[c,a,c,b]).
false
```

**Ex. 6** Escreva um predicado `bissexto(A)` que recebe um ano e é verdadeiro se ele for bissexto ou falso em caso contrário.

```
?- bissexto(2021).
false

?- bissexto(2000).
true

?- bissexto(2004).
true

?- bissexto(1900).
false
```

**Ex. 7** Escreva um predicado `romano(N,R)` que recebe um número natural positivo `N` e devolve uma lista `R` representando o número recebido em numeração romana.

```
?- romano(21, R).
R = ['X', 'X', 'I']

?- romano(800, R).
R = ['D', 'C', 'C', 'C']

?- romano(2021, R).
R = ['M', 'M', 'X', 'X', 'I']
```

**Ex. 8** Considere uma representação de conjuntos como listas. Defina os seguintes predicados:

- (a) `disjunto(L,K)` é verdadeiro se, e somente se, `L` e `K` são disjuntos, ou seja, não possuem elementos em comum.

```
?- disjunto([1,3,5],[2,4,6]).
true

?- disjunto([1,3,5],[2,4,5]).
false
```

- (b) `união(L,K,M)` é verdadeiro se, e somente se, `M` é a união de `L` e `K`.

```
?- união([1,3,5],[2,4,6],[1,3,2,4,6,5]).
true

?- união([1,3,5],[2,4,5],[1,3,2,4,6,5]).
false
```

(c) `interseção(L,K,M)` é verdadeiro se, e somente se, `M` é a interseção de `L` e `K`.

```
?- interseção([1,3,5],[2,4,6],[]).  
true  
  
?- interseção([1,3,5],[2,4,5],[5]).  
true  
  
?- interseção([1,3,5],[2,4,5],[5,6]).  
false
```

(d) `diferença(L,K,M)` é verdadeiro se, e somente se, `M` é a diferença entre `L` e `K`.

```
?- diferença([1,3,5],[2,4,6],[1,3,5]).  
true  
  
?- diferença([1,3,5],[2,4,5],[1,3]).  
true  
  
?- diferença([1,3,5],[1,4,5],[4]).  
false
```

**Ex. 9** Defina um predicado `ocorrências(X,L,N)` que é verdadeiro se, e somente se, `X` ocorre `N` vezes na lista `L`.

**Ex. 10** Defina um predicado `ocorre(L,N,X)` que é verdadeiro se, e somente se, `X` é o elemento que ocorre na posição `N` da lista `L`.

**Ex. 11** Escreva um predicado `fatores_primos(N,F)` que recebe um número natural positivo `N` e devolve uma lista `F` contendo sua decomposição em fatores primos.

```
?- fatores_primos(1, F).  
F = []  
  
?- fatores_primos(2, F).  
F = [2]  
  
?- fatores_primos(9, F).  
F = [3, 3]  
  
?- fatores_primos(12, F).  
F = [2, 2, 3]  
  
?- fatores_primos(901255, F).  
F = [5, 17, 23, 461]
```

## 2 Sugestões de leitura

- Leia os capítulos do livro de Eloi Favero referentes ao tema discutido nesta prática.

- Leia o wikilivro sobre Prolog em <http://pt.wikibooks.org/wiki/Prolog>.