# Capítulo 7

# Prolog (Listas)

## 7.6 Listas

7.6.1. Defina o predicado `nao_membro/2`, tal que `nao_membro(El, Lst)`, em que `El` é um elemento e `Lst` é uma lista, significa que `El` não pertence a `Lst`. Por exemplo,

```
?- nao_membro(8, [1,a,2,8,c]).
false.

?- nao_membro(8, [1,a,2,9,c]).
true
false.

?- nao_membro(8, [1,a,2,X,c]).
true .

?- nao_membro(Z, [1,a,2,X,c]).
true.

?- nao_membro(X, [1,a,2,X,c]).
false.
```

7.6.2. Defina o predicado `insere_ordenado/3`, tal que `insere_ordenado(El, Lst1, Lst2)`, em que `El` é um inteiro e `Lst1` e `Lst2` são listas de inteiros ordenadas de forma ascendente, significa que `Lst2` é o resultado de inserir `El` em `Lst1`. Por exemplo,

```
?- insere_ordenado(5, [1,2,3,6,9], L).
L = [1, 2, 3, 5, 6, 9] .

?- insere_ordenado(0, [1,2,3,6,9], L).
L = [0, 1, 2, 3, 6, 9] .

?- insere_ordenado(10, [1,2,3,6,9], L).
L = [1, 2, 3, 6, 9, 10] .

?- insere_ordenado(3, [1,2,3,6,9], L).
L = [1, 2, 3, 3, 6, 9] .
```

7.6.3. Defina o predicado `junta_novo_aleatorio/4`, tal que `junta_novo_aleatorio(Lst1, Lim_Inf, Lim_Sup, Lst2)`, em que `Lst1` e `Lst2` são listas de inteiros ordenadas de forma ascendente, e `Lim_Inf` e `Lim_Sup` são dois inteiros, tais que `Lim_Inf` $\leq$ `Lim_Sup`, significa que `Lst2` é o resultado de inserir em `Lst1` um inteiro aleatório `Al`, tal que `Lim_Inf` $\leq$ `Al` $\leq$ `Lim_Sup`, e `Al` não pertence a `Lst1`. Por exemplo,

```
?- junta_novo_aleatorio([1,3], 1, 5, Lst2).
Lst2 = [1, 3, 5] .

?-  junta_novo_aleatorio([1,3], 1, 5, Lst2).
Lst2 = [1, 2, 3] .

?-  junta_novo_aleatorio([1, 2, 3, 4, 5], 1, 5, Lst2).
false.
```

Para gerar um inteiro aleatório entre dois limites, use o predicado `random_between(Lim_Inf, Lim_Sup, N)`. Sugestão: use os predicados `nao_membro` e `insere_ordenado` definidos anteriormente. NOTA: não se preocupe se o seu predicado por vezes devolver `false` em situações em que não o devia fazer. Este problema será resolvido numa próxima aula.

7.6.4. Defina o predicado `n_aleatorios/4`, tal que `n_aleatorios(N, Lim_Inf, Lim_Sup, Lst)` significa que a lista `Lst` contém `N` inteiros gerados aleatoriamente, entre `Lim_Inf` e `Lim_Sup`, e está ordenada por ordem ascendente. Por exemplo,

```
?- n_aleatorios(3, 1, 5, Lst).
```

```
Lst = [2, 4, 5] .

?- n_aleatorios(3, 1, 5, Lst).
Lst = [1, 3, 5] .

?- n_aleatorios(3, 1, 5, Lst).
Lst = [2, 3, 5] .
```

Sugestão: use o predicado `junta_novo_aleatorio` definido anteriormente.

7.6.5. Defina o predicado `chave_euromilhoes/2`, tal que `chave_euromilhoes(Numeros, Estrelas)` significa que a lista `Numeros` contém 5 inteiros gerados aleatoriamente, entre 1 e 50, e a lista `Estrelas` contém 2 inteiros gerados aleatoriamente, entre 1 e 12. Ambas as listas estão ordenadas por ordem ascendente. Por exemplo,

```
?- chave_euromilhoes(Numeros, Estrelas) .
Numeros = [20, 27, 28, 29, 36],
Estrelas = [10, 12] .

?- chave_euromilhoes(Numeros, Estrelas) .
Numeros = [3, 17, 21, 43, 44],
Estrelas = [5, 8] .

?- chave_euromilhoes(Numeros, Estrelas) .
Numeros = [1, 18, 26, 33, 43],
Estrelas = [5, 6] .
```

Sugestão: use o predicado `n_aleatorios` definido anteriormente.

7.6.6. Defina o predicado `comp_maior_lista/2`, tal que `comp_maior_lista(L, C)`, em que `L` é uma lista de listas, significa que o maior comprimento das listas de `L` é `C`. Por exemplo,

```
?- comp_maior_lista([[1,2,3], [4,3,1,3], []], C).
C = 4
```

Sugestão: use o predicado `length(Lst, Comp)`.

7.6.7. Defina o predicado `duplica_elementos/2`, tal que `duplica_elementos(Lst1, Lst2)` significa que a lista `Lst2` é o resultado de repetir cada um dos elementos da lista `Lst1`. Por exemplo,

```
?- duplica_elementos([a,b,c], L).
L = [a, a, b, b, c, c].

?- duplica_elementos([], L).
L = [].
```

(a) gerando um processo recursivo.

(b) gerando um processo iterativo.