



Universidade Federal da Bahia

Instituto de Computação - DCC/UFBA



Aula 07 - PROLOG - Recursão



Disciplina: MATA56 - Paradigmas de Linguagens de Programação

Profº Claudio Junior Nascimento da Silva
claudiojns@ufba.br

Agenda

- Recursão:
 - Conceitos;
 - Estratégia;
 - Definição;
 - Exemplos.
- Exercícios:
 - Resolução de exercícios em sala de aula;
 - Envio para e-mail (claudiojr.professor@gmail.com) até 30/05/2023;

Recursão

- Mecanismo de programação no qual uma definição de função ou de outro objeto refere-se ao próprio objeto;
- É uma função que é definida em termos de si mesma;
- Mecanismo básico para repetições nas linguagens funcionais;
- Recursão, recursividade, recorrência;

Estratégia

1. Dividir o problema em problemas menores do mesmo tipo;
 2. Resolver os problemas menores (dividindo-os em problemas ainda menores, se necessário);
 3. Combinar as soluções dos problemas menores para formar a solução final;
- Ao dividir o problema sucessivamente em problemas menores, eventualmente os caso simples são alcançados:
 - Não podem ser divididos;
 - Suas soluções são definidas explicitamente

Definição recursiva

1. Há um ou mais casos base que dizem o que fazer em situações simples, onde não é necessária nenhuma recursão:
 1. Resposta imediata;
 2. Não é necessário chamar recursivamente a função;
 3. Garante, eventualmente, que a recursão pode parar;
2. Há um ou mais casos recursivos que são mais gerais, e definem a função em termos de uma chamada mais simples a si mesma.

Exemplo 01 - Fatorial

Fatorial N:

| $N = 0, 1$

| $N > 0, N * \text{Fatorial}(N-1).$

Fatorial 5:

$\Rightarrow 5 * \text{Fatorial}(4) \Rightarrow 5 * 24 = 120$

$\Rightarrow \text{Fatorial}(4) : 4 * \text{Fatorial}(3) \Rightarrow 4 * 6 = 24$

$\Rightarrow \text{Fatorial}(3) : 3 * \text{Fatorial}(2) \Rightarrow 3 * 2 = 6$

$\Rightarrow \text{Fatorial}(2) : 2 * \text{Fatorial}(1) \Rightarrow 2 * 1 = 2$

$\Rightarrow \text{Fatorial}(1) : 1 * \text{Fatorial}(0) \Rightarrow 1 * 1 = 1$

$\Rightarrow \text{Fatorial}(0) : 1$

$5 * (4 * (3 * (2 * (1 * 1))))$

$5 * (4 * (3 * (2 * 1)))$

$5 * (4 * (3 * 2))$

$5 * (4 * 6)$

$5 * 24$

120

Exemplo 01 - Fatorial

```
fatorial(0,1).      % caso base


fatorial(N,F) :-    % demais casos
    N > 0,          % se N > 0, senão caso base
    N1 is N - 1,    % N1 = N - 1
    fatorial(N1, F1), % chamada recursiva
    F is N * F1.     % unifica F com N * F1
```

Exemplo 01 - Fatorial

 `fatorial(10,F).`

F = 3628800

Next 10 100 1,000 Stop

 `fatorial(10000,F).`

F =

284625968091705451890641321211986889014805140170279923079417999427441134000376444377

0.530 seconds cpu time

Next 10 100 1,000 Stop

 `fatorial(100000,F).`

F =

282422940796034787429342157802453551847749492609122485057891808654297795090106301787

44.618 seconds cpu time

Next 10 100 1,000 Stop 

```
fatorial(0, 1).  
fatorial(N, F) :-  
    N > 0, N1 is N -  
    fatorial(N1, F1),  
    F is N * F1.
```


Exemplo 01 - Fatorial



`fatorial(1000000,F).`



Stack limit (0.2Gb) exceeded

Stack sizes: local: 0.2Gb, global: 44.2Mb, trail: 7.7Mb

Stack depth: 996,805, last-call: 0%, Choice points: 1,000,019

In:

[996,805] system:is(_1716, <compound (*)/2>)

[996,804] fatorial(3225, _1750)

[996,803] fatorial(3226, _1776)

[996,802] fatorial(3227, _1802)

[996,801] fatorial(3228, _1828)

Use the `--stack_limit=size[KMG]` command line option or

?- `set_prolog_flag(stack_limit, 2_147_483_648).` to double the limit.

?-

`fatorial(1000000,F).`

Exemplo 02 - Potenciação

1. Crie um programa Prolog para calcular o **Resultado** de um número **N** elevado a uma potência **E**:
 - 1.O usuário deverá informar o número (N). Caso digite (-1) o programa será encerrado;
 - 2.O usuário deverá informar o expoente (E), que deverá ser ≥ 0 ;
 - 3.O sistema deverá usar Recursão para calcular o Resultado.

Exemplo 02 - Potenciação

`power(_, 0, 1).`

Qual o caso base?

`power(N, E, Resultado) :-`

`E > 0, E1 is E - 1,`

`power(N, E1, Parcial),`

`Resultado is N * Parcial.`

`start :-`

`write('Informe um número (-1 para sair): ', read(N),
(N == -1 -> true ; calcular_potencia(N)).`

`calcular_potencia :-`

`write('Informe o expoente: '), read(E),
power(N, E, Resultado),
write('O resultado: '), write(Resultado), nl,
start.`

```
power(_, 0, 1).
```

```
power(N, E, Resultado) :-  
    E > 0,  
    E1 is E - 1,  
    power(N, E1, Parcial),  
    Resultado is N * Parcial.
```

```
start :-  
    write('Informe um número (-1 para sair): '),  
    read(N),  
    (N == -1 -> true ; calcular_potencia(N)).
```

```
calcular_potencia(N) :-  
    write('Informe o expoente: '),  
    read(E),  
    power(N, E, Resultado),  
    write('O resultado: '),  
    write(Resultado),  
    nl,  
    start.
```



start.

Informe um número (-1 para sair):

144

Informe o expoente:

12

O resultado: 79496847203390844133441536

Informe um número (-1 para sair):

12

?-

start.



Vamos aos Exercícios