



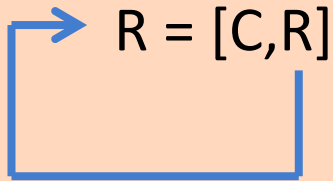
PROGRAMAÇÃO ESTRUTURADA

Recursividade

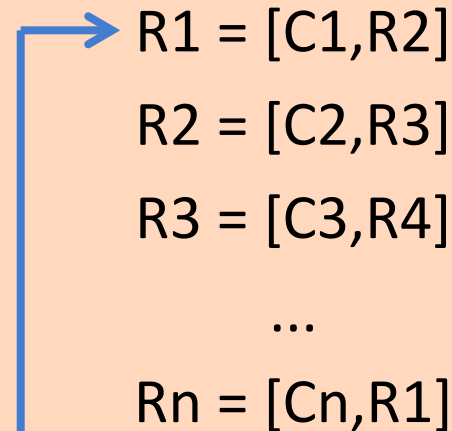
RECURSIVIDADE

- Quando uma função chama a si própria direta ou indiretamente é caracterizada a recursividade

RECURSÃO DIRETA



RECURSÃO INDIRETA



RECURSIVIDADE

- Todo algoritmo deve ser executado em um número finito de passos → tempo finito
- Logo, para garantir que uma chamada recursiva não crie um loop infinito é necessário que ela esteja subordinada a uma expressão lógica que, em algum instante, se tornará falsa
 - Isto permitirá que a execução termine

EXEMPLO - RECURSIVIDADE

```
#include <stdio.h>

/* AQUI DEFINIREMOS A FUNÇÃO PRODUTO */

int main (void) {
    int num;
    do {
        printf("Digite um número inteiro positivo:");
        scanf("%d", &num);
    } while (num <= 0);
    if (num % 2 == 0)
        num--;
    printf("O produto é: %d", produto(num));
    return 0;
}
```

Calcular o produto
dos números
inteiros positivos
ímpares menores
ou iguais a N

EXEMPLO - RECURSIVIDADE

```
int produto (int N) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(N-2) * N);  
}
```

Vamos supor que o usuário digite $N = 7$

A resposta será: $1 \times 3 \times 5 \times 7 = 105$

Na função main :

```
printf("O produto é: %d", produto(7));
```

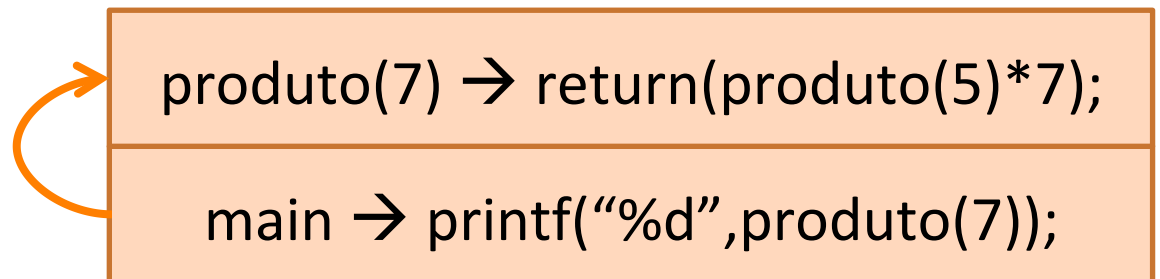
EXEMPLO – PASSO A PASSO

```
int produto (N=7) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(5) * 7);  
}
```

main → printf("%d",produto(7));

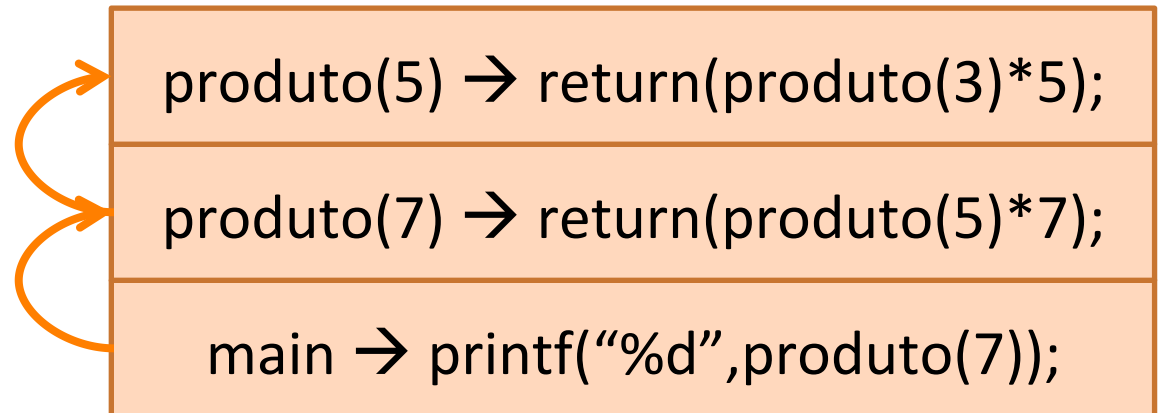
EXEMPLO – PASSO A PASSO

```
int produto (N=7) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(5) * 7);  
}
```



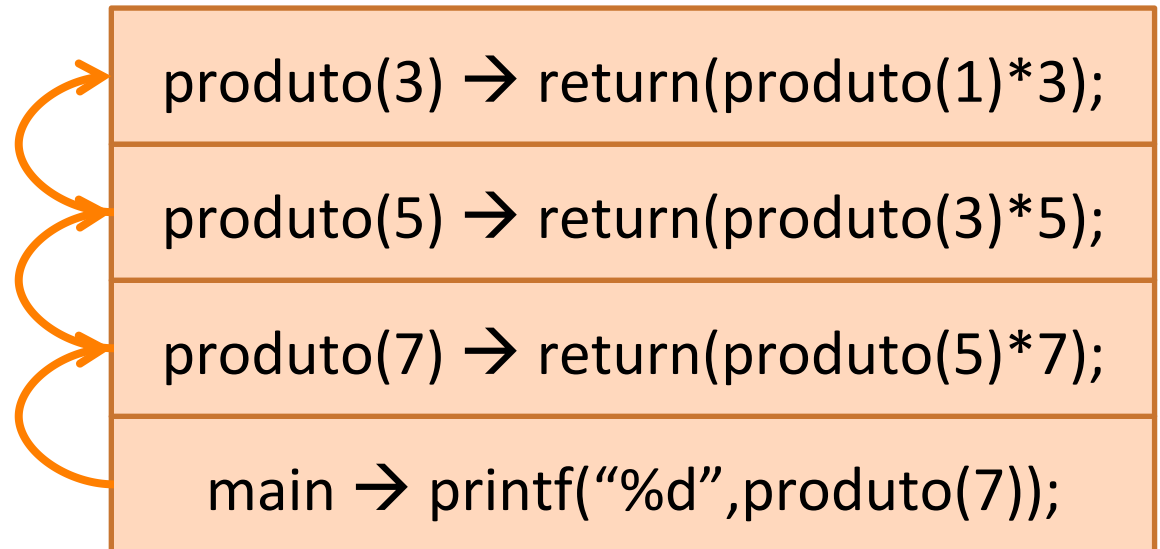
EXEMPLO – PASSO A PASSO

```
int produto (N=5) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(3) * 5);  
}
```



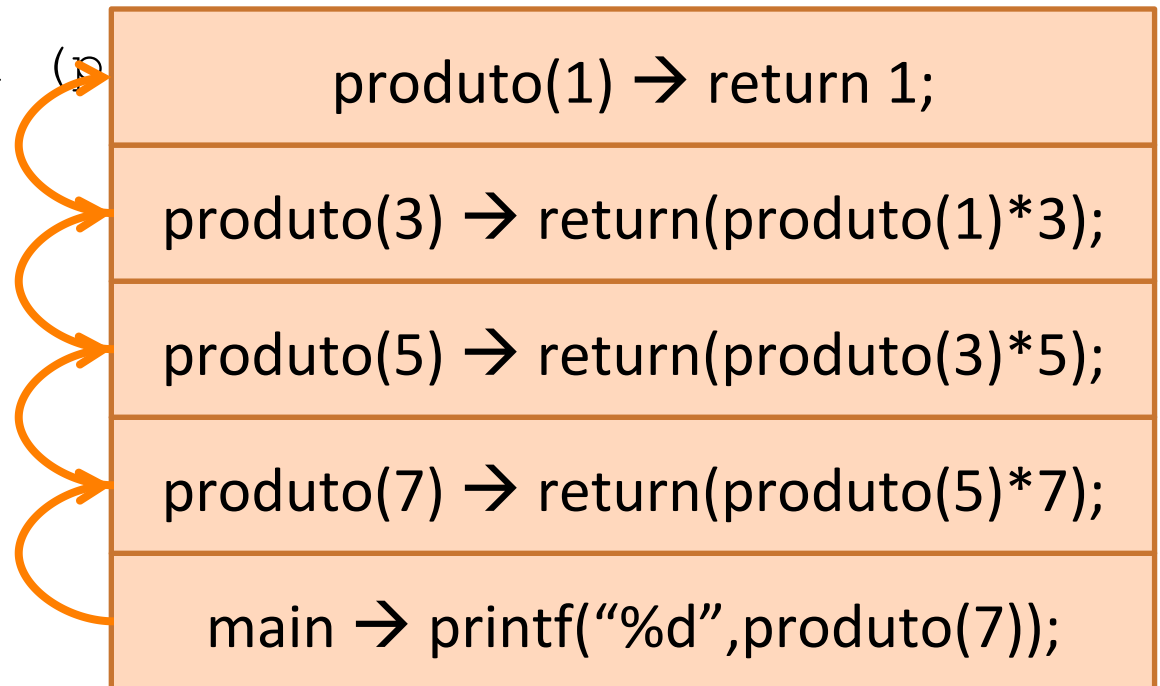
EXEMPLO – PASSO A PASSO

```
int produto (N=3) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(1) * 3);  
}
```



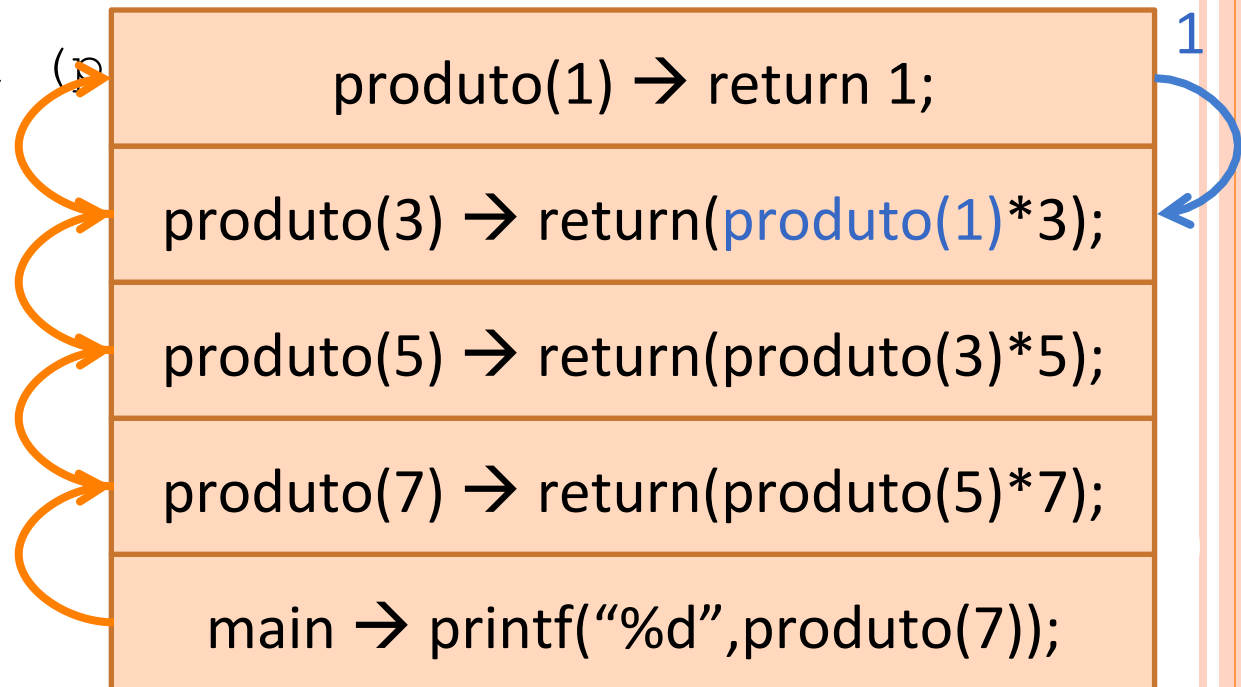
EXEMPLO – PASSO A PASSO

```
int produto (N=1) {  
    if (N == 1)  
        return 1;  
    else  
        return (p  
}  
}
```



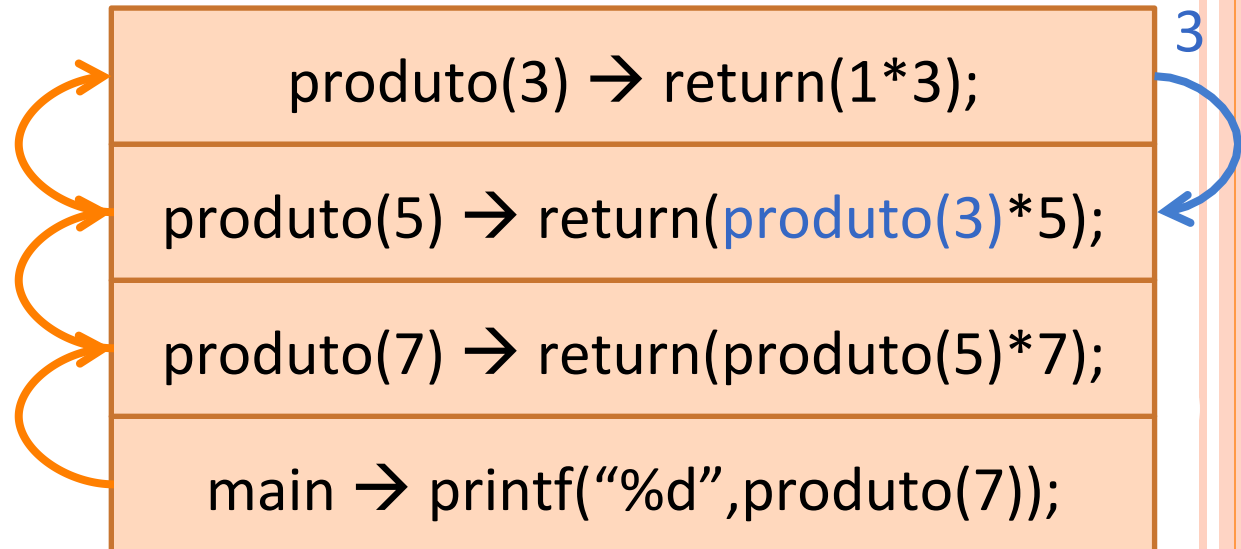
EXEMPLO – PASSO A PASSO

```
int produto (N=1) {  
    if (N == 1)  
        return 1;  
    else  
        return (p  
}  
}
```



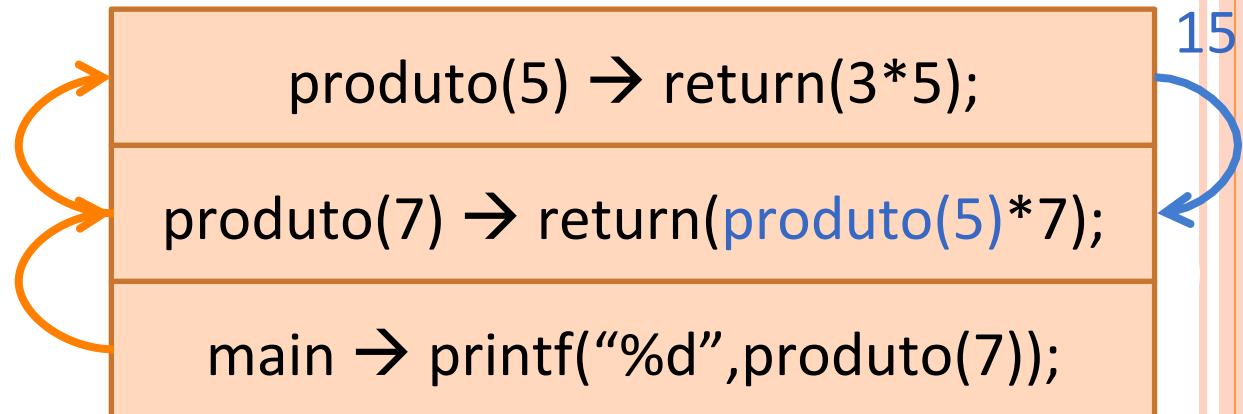
EXEMPLO – PASSO A PASSO

```
int produto (N) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(N-2) * N);  
}
```



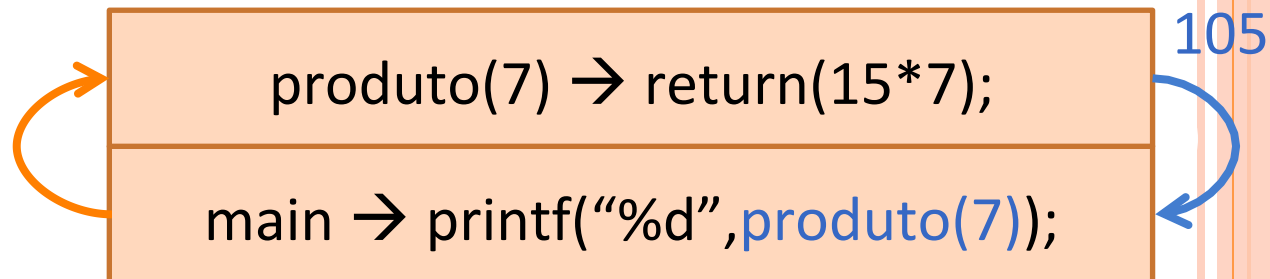
EXEMPLO – PASSO A PASSO

```
int produto (N) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(N-2) * N);  
}
```



EXEMPLO – PASSO A PASSO

```
int produto (N) {  
    if (N == 1)  
        return 1;  
    else  
        return (produto(N-2) * N);  
}
```



EXEMPLO – PASSO A PASSO

```
int main (void) {  
    int num;  
    do {  
        printf("Digite um número inteiro positivo:");  
        scanf("%d", &num);  
    } while (num <= 0);  
    if (num % 2 == 0)  
        num--;  
    printf("O produto é: %d", produto(num));  
    return 0;  
}
```

main → printf("%d",105);

RECURSIVIDADE

- Na prática, ao se definir uma rotina recursiva, dividimos o problema da seguinte forma:
 - **SOLUÇÃO TRIVIAL**: é dada por definição, não precisa de recursividade para ser resolvida
 - **SOLUÇÃO GERAL**: será parte do problema, cuja solução depende da solução do mesmo problema, só que para um caso mais simples ou reduzido

RECURSIVIDADE - EXEMPLO

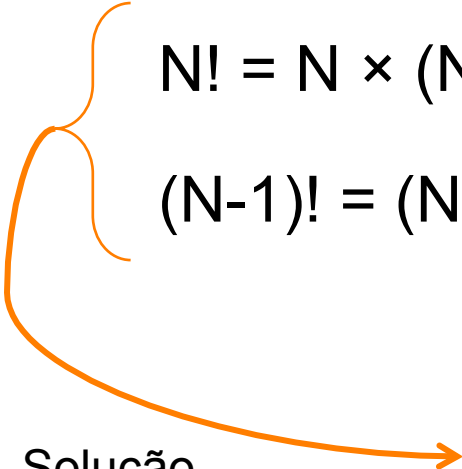
- Considere o problema de se encontrar o fatorial de um número N.

$$N! = N \times (N-1) \times (N-2) \times (N-3) \times \dots \times 1 \times 0!$$

$$(N-1)! = (N-1) \times (N-2) \times (N-3) \times \dots \times 1 \times 0!$$

RECURSIVIDADE - EXEMPLO

- Considere o problema de se encontrar o fatorial de um número N.


$$N! = N \times (N-1) \times (N-2) \times (N-3) \times \dots \times 1 \times 0!$$

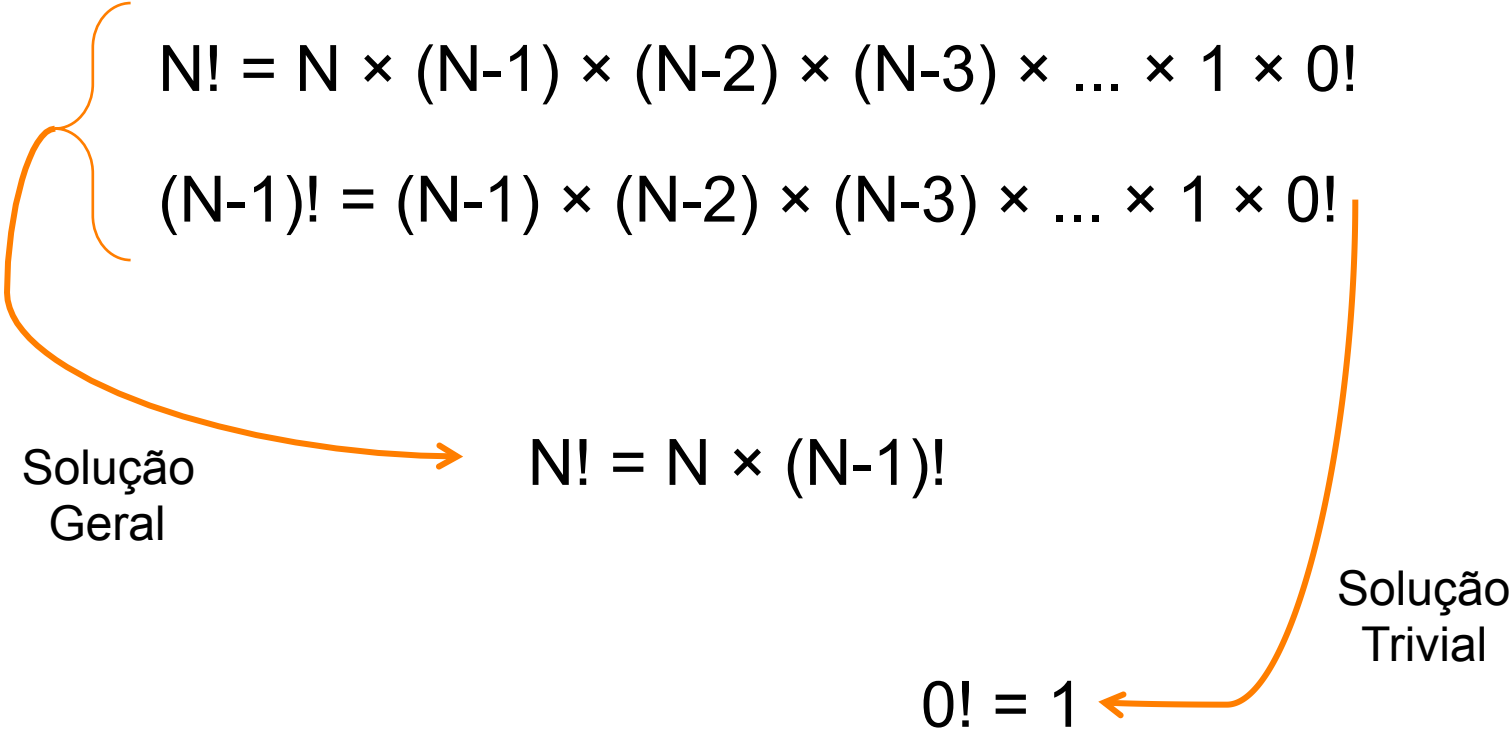
$$(N-1)! = (N-1) \times (N-2) \times (N-3) \times \dots \times 1 \times 0!$$

Solução
Geral

$$N! = N \times (N-1)!$$

RECURSIVIDADE - EXEMPLO

- Considere o problema de se encontrar o fatorial de um número N .


$$N! = N \times (N-1) \times (N-2) \times (N-3) \times \dots \times 1 \times 0!$$

$$(N-1)! = (N-1) \times (N-2) \times (N-3) \times \dots \times 1 \times 0!$$

Solução
Geral

$$N! = N \times (N-1)!$$

Solução
Trivial

$$0! = 1$$

RECURSIVIDADE - FATORIAL

- Solução Geral:

$$\text{FAT}(N) = N \times \text{FAT}(N-1)$$

- Solução Trivial:

$$\text{FAT}(0) = 1$$

RECURSIVIDADE - FATORIAL

- Solução Geral:

$$\text{FAT}(N) = N \times \text{FAT}(N-1)$$

- Solução Trivial:

$$\text{FAT}(0) = 1$$

```
int fatorial(int N) {  
    if (N == 0)  
        return 1;  
    else  
        return (N * fatorial(N-1));  
}
```

EXERCÍCIOS USANDO RECURSIVIDADE

1. Faça uma função que calcule X^n , sendo $N \geq 0$.
2. Faça uma função que some os números inteiros positivos compreendidos entre A e B. Considere que $A < B$ e que A e B devem entrar na soma.
3. Faça uma função que calcule o quociente da divisão de A por B, utilizando subtrações sucessivas.
4. Faça uma função que escreva os números pares menores ou iguais a N
5. Faça uma função que retorne o enésimo termo de uma série de Fibonacci.
 - Pergunta: A recursividade sempre vale a pena?

SOLUÇÃO – EXERCÍCIO 5 – FIBONACCI

○ Solução Trivial:

- Por definição temos que o primeiro termo da série de Fibonnaci é 0 e o segundo termo da série é 1.

$$\text{FIB}(1) = 0$$

$$\text{FIB}(2) = 1$$

○ Solução Geral:

- Um termo N da série de fibonacci é encontrado somado os 2 termos anteriores

$$\text{FIB}(3) = \text{FIB}(2) + \text{FIB}(1) \rightarrow 0 \ 1 \ 1$$

$$\text{FIB}(4) = \text{FIB}(3) + \text{FIB}(2) \rightarrow 0 \ 1 \ 1 \ 2$$

$$\text{FIB}(5) = \text{FIB}(4) + \text{FIB}(3) \rightarrow 0 \ 1 \ 1 \ 2 \ 3$$

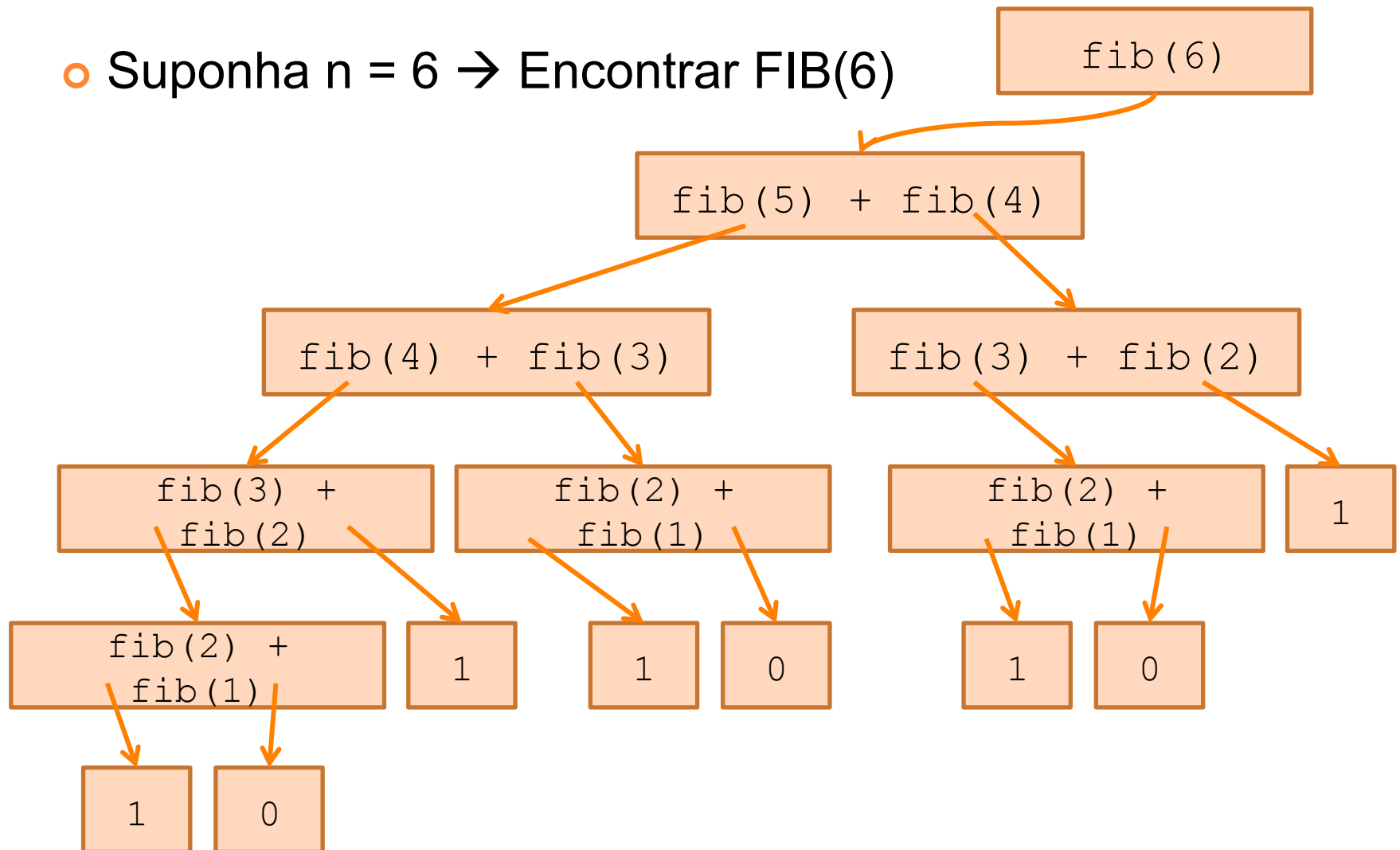
$$\text{FIB}(N) = \text{FIB}(N-1) + \text{FIB}(N-2)$$

SOLUÇÃO – EXERCÍCIO 5 – FIBONACCI

```
int fib(int N) {  
    if (N == 1)  
        return 0;  
    else  
        if (N == 2)  
            return 1;  
        else  
            return (fib(N-2) + fib(N-1));  
}
```

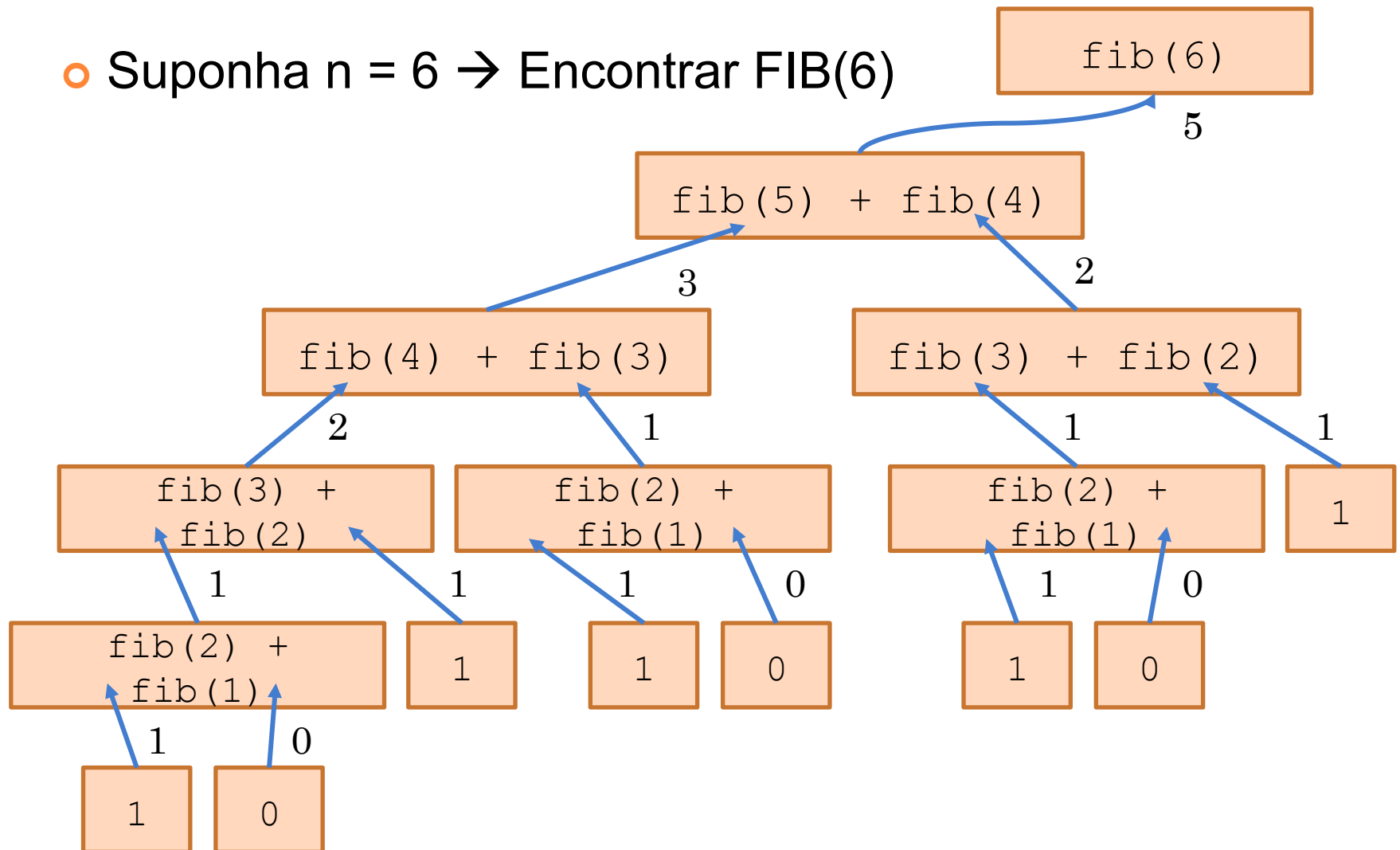

SOLUÇÃO – EXERCÍCIO 5 – FIBONACCI

- Suponha $n = 6 \rightarrow$ Encontrar $FIB(6)$



SOLUÇÃO – EXERCÍCIO 5 – FIBONACCI

- Suponha $n = 6 \rightarrow$ Encontrar $FIB(6)$



SOLUÇÃO – FIBONACCI ITERATIVO

```
int fib(int N) {  
    int T1,T2,T3;  
    if (N == 1)  
        return 0;  
    else  
        if (N == 2)  
            return 1;  
        else {  
            T1 = 0; T2 = 1;  
            for (i = 3; i <= N; i++) {  
                T3 = T1 + T2;  
                T1 = T2;  
                T2 = T3;  
            }  
            return (T3);  
        }  
}
```