Algoritmos e Estruturas de Dados I

Profª Priscilla Abreu

priscilla.abreu@ime.uerj.br

2022.1

Algoritmos e Estruturas de Dados I

Roteiro da aula

- Listas: introdução
- Recursividade

Recursividade

Algoritmos e Estruturas de Dados I

Motivação

Se um problema pode ser resolvido facilmente, **resolva o problema!**

Se o problema é grande...

- Elabore uma solução menor do problema;
- Relacione com o problema maior;
- Resolva o problema menor;
- Volte ao problema inicial.

Algoritmos e Estruturas de Dados I

Uma função pode ser implementada de forma **iterativa** ou **recursiva**.

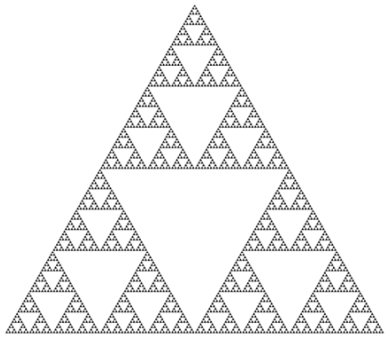
Quase sempre a forma recursiva apresenta uma codificação mais simples (reduzida).

Por outro lado, implementações iterativas tendem a ser mais eficientes (performance) que as recursivas.

Algoritmos e Estruturas de Dados I

Um objeto é denominado recursivo quando sua definição é parcialmente feita em termos dele mesmo.

Em programação, a recursividade é um mecanismo útil e poderoso que permite a uma função chamar a si mesma direta ou indiretamente.



Algoritmos e Estruturas de Dados I

A ideia básica de um algoritmo recursivo:

- Diminuir sucessivamente o problema em subproblemas menores ou mais simples, até chegar a um problema pequeno o suficiente para que ele possa ser resolvido trivialmente.

- A recursão envolve uma função que chama a si mesma.

- Para todo algoritmo recursivo existe um outro correspondente iterativo (não recursivo), que executa a mesma tarefa.

Algoritmos e Estruturas de Dados I

CARACTERÍSTICAS:

- Um algoritmo recursivo deve ter um **caso básico**;
- Um algoritmo recursivo deve mudar o seu estado e se aproximar do caso básico;
- Um algoritmo recursivo deve chamar a si mesmo, recursivamente.

Algoritmos e Estruturas de Dados I

Solução trivial ou caso base: dada por definição; isto é, não necessita da recursão para ser obtida.

Solução geral: parte do problema que em essência é igual ao problema original, sendo porém menor. A solução, neste caso, pode ser obtida por uma chamada recursiva.

Algoritmos e Estruturas de Dados I

Exemplo iterativo – Fatorial :

Construa um programa que, a partir de um número positivo, implemente uma função para calcular e retornar o fatorial deste número.

$$n! = n * (n-1) * (n-2) * ... * 1$$

Algoritmos e Estruturas de Dados I

Exemplo iterativo – Fatorial :

```
#include<stdio.h>
int fatorial(int n){
    int fat =n, i;
    for(i=n-1; i>1; i++){
        fat = fat * i;
    }
    return fat;
}
```

```
função fatorial(n: inteiro): inteiro
inicio
        fat, i: inteiro
        para i:= n-1 até 1 faça
                fat:= fat * i
        fim-para
        retorne fat
fim
```

Algoritmos e Estruturas de Dados I

Exemplo iterativo – Fatorial :

```
int main(){
        int num;
        printf("Informe o número:\n");
        scanf("%d",&num);
        printf("Fatorial de %d é: %d", num, fatorial(num));
}
```

Algoritmos e Estruturas de Dados I

Exemplo fatorial – recursivo:

fatorial(4) = 4 * 3 * 2 * 1

fatorial(3)

fatorial(4) = 4 * fatorial(3)

fatorial(3) = 3 * fatorial(2)

fatorial(2) = 2 * fatorial(1)

fatorial(1) = 1 * fatorial(0)

fatorial(0) = 1

Algoritmos e Estruturas de Dados I

Exemplo fatorial – recursivo:

fatorial(4) = 4 * fatorial(3)

fatorial(3) = 3 * fatorial(2)

fatorial(2) = 2 * fatorial(1)

fatorial(1) = 1 * 1

fatorial(0) = 1

Algoritmos e Estruturas de Dados I

Exemplo fatorial – recursivo:

fatorial(4) = 4 * fatorial(3)

fatorial(3) = 3 * fatorial(2)

fatorial(2) = 2 * 1

fatorial(1) = 1 * 1

fatorial(0) = 1

Algoritmos e Estruturas de Dados I

Exemplo fatorial – recursivo:

fatorial(4) = 4 * fatorial(3)

fatorial(3) = 3 * 2

fatorial(2) = 2 * 1

fatorial(1) = 1 * 1

fatorial(0) = 1

Algoritmos e Estruturas de Dados I

Exemplo fatorial – recursivo:

fatorial(4) = 4 * 6

fatorial(3) = 3 * 2

fatorial(2) = 2 * 1

fatorial(1) = 1 * 1

fatorial(0) = 1

Algoritmos e Estruturas de Dados I

Exemplo fatorial – recursivo:

fatorial(4) = 4 * 6

fatorial(3) = 3 * 2

fatorial(2) = 2 * 1

fatorial(1) = 1 * 1

fatorial(0) = 1

fatorial(4) = 24

Algoritmos e Estruturas de Dados I

Fatorial – recursivo:

Alternativamente, o fatorial pode ser definido como o produto deste número pelo fatorial de seu predecessor, ou seja:

$$n! = n * (n - 1)!$$

fatorial(n) → n * fatorial(n-1)

Algoritmos e Estruturas de Dados I

Fatorial – recursivo:

Deste modo, podemos escrever uma função recursiva em que cada chamada da função que calcula o fatorial chama a própria função fatorial.

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

```
função fatorial(n: inteiro): inteiro
inicio
   se (n = 0) então
        retorne 1
   senão
        retorne (n* fatorial(n-1))
fim
```

Algoritmos e Estruturas de Dados I

Fatorial – recursivo:

```
int main(){
        int num;
        printf("Informe o número:\n");
        scanf("%d",&num);
   printf("Fatorial de %d é: %d", num, fatorial(num));
}
```

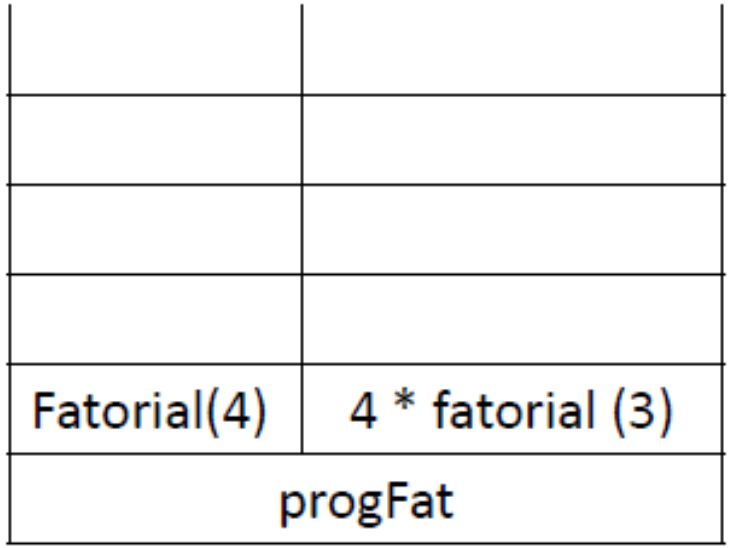
Algoritmos e Estruturas de Dados I

Para executar uma função recursiva, o computador deverá resolver todas as chamadas recursivas antes de retornar um valor na primeira chamada da função.

Isto gera uma série de chamadas que, se corretamente implementadas, terminará retornando o valor especificado no caso básico da definição da função (caso em que n = 0, para a função fatorial).

Algoritmos e Estruturas de Dados I

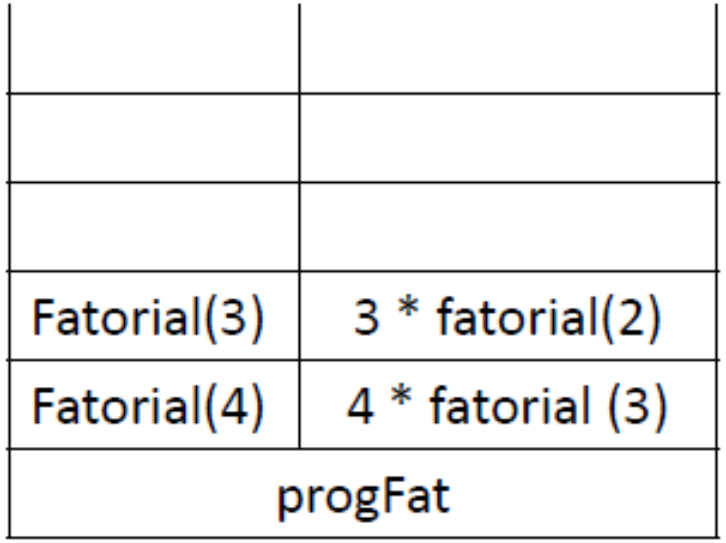
Simulação: Fatorial(4)




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

Algoritmos e Estruturas de Dados I

Simulação: Fatorial(4)




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

Algoritmos e Estruturas de Dados I

Simulação: Fatorial(4)

Fatorial(2)	2 * fatorial(1)
Fatorial(3)	3 * fatorial(2)
Fatorial(4)	4 * fatorial (3)
progFat	

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

Algoritmos e Estruturas de Dados I

Simulação: Fatorial(4)

Fatorial(1)	1* fatorial(0)
Fatorial(2)	2 * fatorial(1)
Fatorial(3)	3 * fatorial(2)
Fatorial(4)	4 * fatorial (3)
progFat	

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

Algoritmos e Estruturas de Dados I

Simulação: Fatorial(4)

Fatorial(0)	1
Fatorial(1)	1* fatorial(0)
Fatorial(2)	2 * fatorial(1)
Fatorial(3)	3 * fatorial(2)
Fatorial(4)	4 * fatorial (3)
progFat	

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

Algoritmos e Estruturas de Dados I

Simulação: Fatorial(4)

Fatorial(0)	1
Fatorial(1)	1* 1
Fatorial(2)	2 * fatorial(1)
Fatorial(3)	3 * fatorial(2)
Fatorial(4)	4 * fatorial (3)
progFat	

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

Algoritmos e Estruturas de Dados I

Simulação: Fatorial(4)

Fatorial(1)	1
Fatorial(2)	2 * 1
Fatorial(3)	3 * fatorial(2)
Fatorial(4)	4 * fatorial (3)
progFat	

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

Algoritmos e Estruturas de Dados I

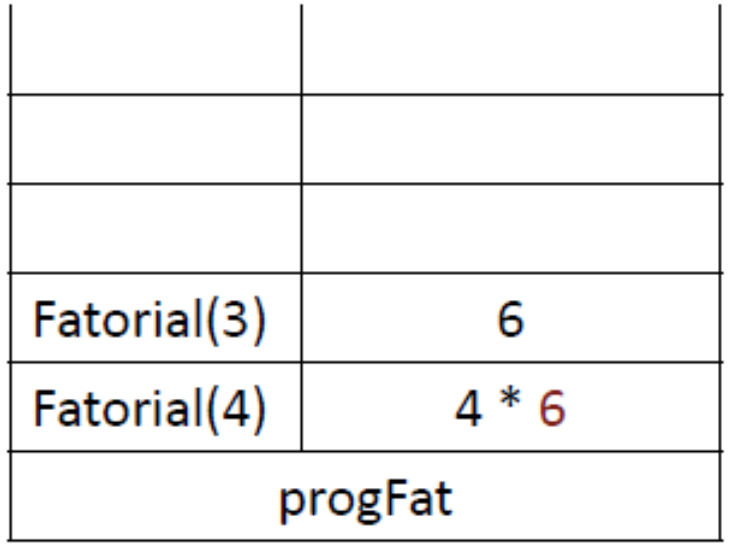
Simulação: Fatorial(4)

Fatorial(2)	2
Fatorial(3)	3 * 2
Fatorial(4)	4 * fatorial (3)
progFat	

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

Algoritmos e Estruturas de Dados I

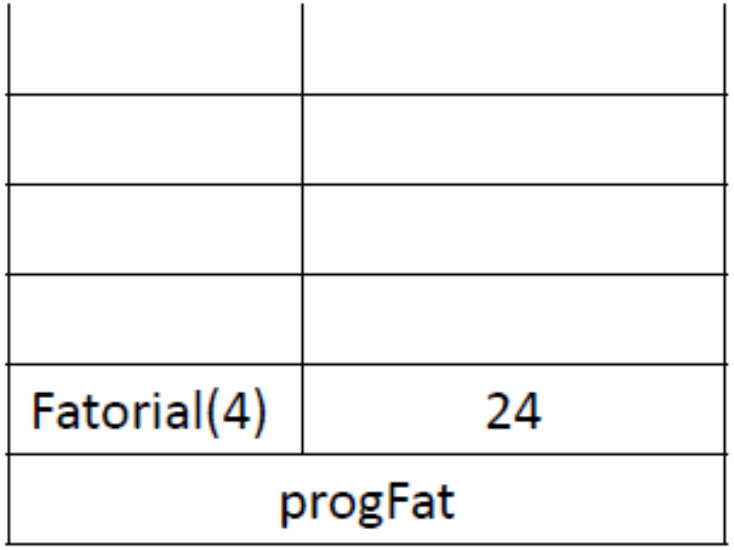
Simulação: Fatorial(4)




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

Algoritmos e Estruturas de Dados I

Simulação: Fatorial(4)




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

Algoritmos e Estruturas de Dados I

EXERCÍCIO

Mostre detalhadamente o resultado do seguinte procedimento para n = 6:

```
void funcao(int n){
        if (n == 0)
                printf(“Zero ”);
        else
        {
                printf(“%d ”,n);
                funcao(n-1);
        }
}
```

Algoritmos e Estruturas de Dados I

Exemplo: Sequência de Fibonacci

Sequência numérica proposta pelo matemático Leonardo Pisa, mais conhecido como Fibonacci. Pode ser definida recursivamente como:

$$F_n = F_{n-1} + F_{n-2}$$

Onde: $F_1 = 1 \; e \; F_2 = 1$

1 1 2 3 5 8 13 21 ...

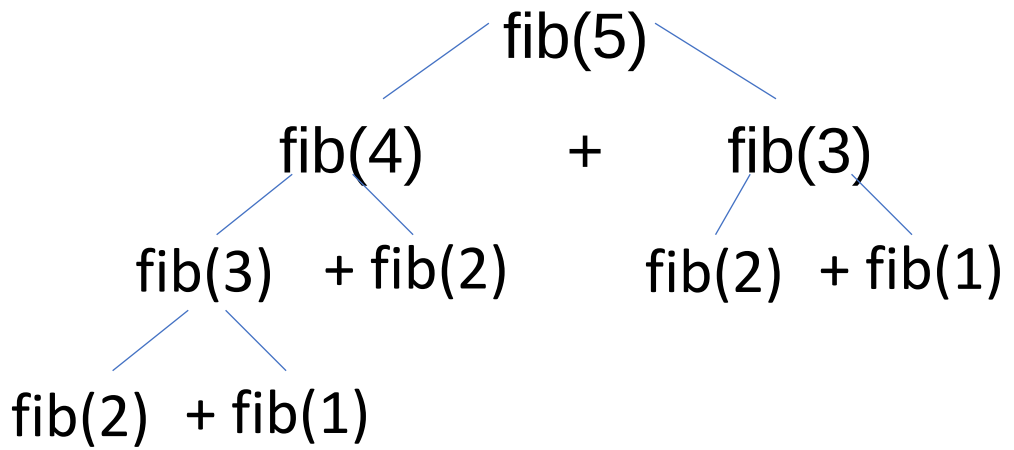
Algoritmos e Estruturas de Dados I

Função para cálculo da sequência de Fibonacci:

```
int fibonacci(int n){
        if ((n ==1) || (n==2))
            return 1;
        else
            return( fibonacci(n – 1) + fibonacci(n – 2));
}
```

Algoritmos e Estruturas de Dados I

Exemplo: árvore recursiva



E se fosse fib(100)???

Algoritmos e Estruturas de Dados I

Vantagens:

- Código mais “enxuto”;
- Mais fácil de ser compreendido e de ser implementado em linguagem de programação;
- Bom para definições matemáticas;

Desvantagens:

- Maior consumo de recursos;
- Tendem a ser mais lentos que os algoritmos iterativos correspondentes;
- Mais difíceis de serem testados no caso de haver muitas chamadas.

Algoritmos e Estruturas de Dados I

Desenvolvendo algoritmos recursivos:

- Ache a condição de parada

Uma condição de parada indica que o problema ou uma parte razoável do mesmo foi feito. Ela é geralmente pequena e dá a solução trivial, sem a necessidade da recursão.

- Ache a solução geral;
  - Como este problema pode ser dividido?
  - Como posso desenvolver o ponto principal da solução geral?
  - Combine a condição de parada e a solução geral.

Algoritmos e Estruturas de Dados I

Desenvolvendo algoritmos recursivos:

- Verifique o término
  - Certificar que a recursão sempre terminará;
  - Comece com uma situação geral;
  - Verifique se em um número finito de passos, a regra de parada será satisfeita e a recursão termina.

- Desenhe uma árvore de recursão
  - É a ferramenta chave para analisar algoritmos recursivos.

Algoritmos e Estruturas de Dados I

Exercício

Determine o que a seguinte função recursiva em C calcula, n>=0.

```
 int func(int n){
            if( n == 0)
                return 0;
            else
                return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(5)	

```
int func(int n){
      if( n == 0)
            return 0;
      return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(5)	5 + FUNC(4)

```
int func(int n){
      if( n == 0)
            return 0;
      return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(4)	
FUNC(5)	**5 + FUNC(4)**

```
int func(int n){
      if( n == 0)
            return 0;
      return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

CHAMADAS	RESULTADO
FUNC(4)	**4 + FUNC(3)**
FUNC(5)	**5 + FUNC(4)**

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(3)	
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
      if( n == 0)
            return 0;
      return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

CHAMADAS	RESULTADO
FUNC(3)	**3 + FUNC(2)**
FUNC(4)	**4 + FUNC(3)**
FUNC(5)	**5 + FUNC(4)**

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(2)	
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(2)	2 + FUNC(1)
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(1)	
FUNC(2)	2 + FUNC(1)
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(1)	1 + FUNC(0)
FUNC(2)	2 + FUNC(1)
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(0)	
FUNC(1)	1 + FUNC(0)
FUNC(2)	2 + FUNC(1)
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(0)	0
FUNC(1)	1 + FUNC(0)
FUNC(2)	2 + FUNC(1)
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(1)	1 + 0
FUNC(2)	2 + FUNC(1)
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(2)	2 + 1 + 0
FUNC(3)	3 + FUNC(2)
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(3)	3 + 2 + 1 + 0
FUNC(4)	4 + FUNC(3)
FUNC(5)	5 + FUNC(4)

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

CHAMADAS	RESULTADO
FUNC(4)	**4 + 3 + 2 + 1 + 0**
FUNC(5)	**5 + FUNC(4)**

Algoritmos e Estruturas de Dados I

Exercício

```
int func(int n){
        if( n == 0)
                return 0;
        return (n+ func(n-1));
}
```

CHAMADAS	RESULTADO
FUNC(5)	**5 +4 +3 +2 +1 +0**

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO
FUNC(5)	**15**

```
int func(int n){
      if( n == 0)
            return 0;
      return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

Exercício

CHAMADAS	RESULTADO

```
int func(int n){
      if( n == 0)
            return 0;
      return (n+ func(n-1));
}
```

Algoritmos e Estruturas de Dados I

EXERCÍCIO – ENTREGA

Implementar uma função recursiva para calcular x^n, com as seguintes características:

$$x^n = x * x^{n-1}$$

Onde:

- $x^0 = 1$
- $n >= 0$

FIM