

ESTRUTURA DE DADOS

Prof.^a Priscilla Abreu

priscilla.braz@rj.senac.br



Estrutura de dados



Roteiro de Aula

- Objetivo da aula
- Revisando função
- Recursividade
- Exercício

Estrutura de dados



Objetivo da aula

Conhecer a técnica de recursividade e implementar funções recursivas.

Estrutura de dados



Função fatorial

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

$$n! = n * (n-1) * (n-2) * \dots * 1$$

Estrutura de dados



Função FATORIAL

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

Estrutura de dados



Função FATORIAL

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

RECURSIVIDADE

Estrutura de dados



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.

Estrutura de dados



Recursividade

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

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

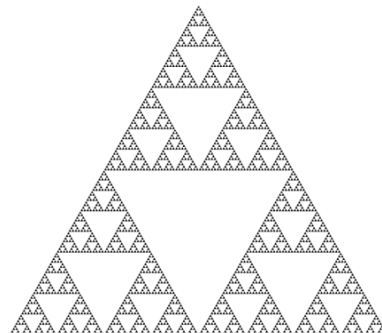
Por outro lado, implementações iterativas tendem a ser mais eficientes (performance) que as recursivas.

Estrutura de dados

Recursividade

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.



Estrutura de dados



Recursividade

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.

Estrutura de dados



Recursividade

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;

Estrutura de dados



Recursividade

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.

Estrutura de dados



Recursividade – exemplo

Voltando ao exemplo do fatorial...

$$\text{fatorial}(4) = 4 * \underbrace{3 * 2 * 1}_{\text{fatorial}(3)}$$

$$\text{fatorial}(4) = 4 * \text{fatorial}(3)$$

$$\text{fatorial}(3) = 3 * \text{fatorial}(2)$$

$$\text{fatorial}(2) = 2 * \text{fatorial}(1)$$

$$\text{fatorial}(1) = 1 * \text{fatorial}(0)$$

Estrutura de dados



Recursividade – exemplo

Voltando ao exemplo do fatorial...

$$\text{fatorial}(4) = 4 * \text{fatorial}(3)$$

$$\text{fatorial}(3) = 3 * \text{fatorial}(2)$$

$$\text{fatorial}(2) = 2 * \text{fatorial}(1)$$

$$\text{fatorial}(1) = 1 * 1$$

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

Estrutura de dados



Recursividade – exemplo

Voltando ao exemplo do fatorial...

$$\text{fatorial}(4) = 4 * \text{fatorial}(3)$$

$$\text{fatorial}(3) = 3 * \text{fatorial}(2)$$

$$\text{fatorial}(2) = 2 * 1$$

$$\text{fatorial}(1) = 1 * 1$$

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

Estrutura de dados



Recursividade – exemplo

Voltando ao exemplo do fatorial...

$$\text{fatorial}(4) = 4 * \text{fatorial}(3)$$

$$\text{fatorial}(3) = 3 * 2$$

$$\text{fatorial}(2) = 2 * 1$$

$$\text{fatorial}(1) = 1 * 1$$

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

Estrutura de dados



Recursividade – exemplo

Voltando ao exemplo do fatorial...

$$\text{fatorial}(4) = 4 * 6$$

$$\text{fatorial}(3) = 3 * 2$$

$$\text{fatorial}(2) = 2 * 1$$

$$\text{fatorial}(1) = 1 * 1$$

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

Estrutura de dados



Recursividade – exemplo

Voltando ao exemplo do fatorial...

$$\text{fatorial}(4) = 4 * 6$$

$$\text{fatorial}(3) = 3 * 2$$

$$\text{fatorial}(2) = 2 * 1$$

$$\text{fatorial}(1) = 1 * 1$$

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

$$\text{fatorial}(4) = 24$$

Estrutura de dados



Recursividade – exemplo

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

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

$$\text{fatorial}(n) \rightarrow n * \text{fatorial}(n-1)$$

Estrutura de dados



Recursividade – exemplo

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));  
}
```

Estrutura de dados



Função FATORIAL

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

Recursividade

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).

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: Fatorial(4)**

Fatorial(4)	4 * fatorial (3)
progFat	

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


Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: Fatorial(4)**

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));  
}
```

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: 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));  
}
```

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: 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));  
}
```

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: 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));  
}
```

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: 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));  
}
```

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: 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));  
}
```

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: 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));  
}
```

Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: Fatorial(4)**

Fatorial(3)	6
Fatorial(4)	4 * 6
progFat	

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


Estrutura de dados



Recursividade – exemplo

**Simulação do uso do exemplo anterior para 4!.
Programa Fat: Fatorial(4)**

Fatorial(4)	24
progFat	

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

Estrutura de dados



Recursividade – 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);  
    }  
}
```

Recursividade – exemplo

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 ...

Estrutura de dados



Recursividade – exemplo

Função para cálculo da sequência de Fibonacci:

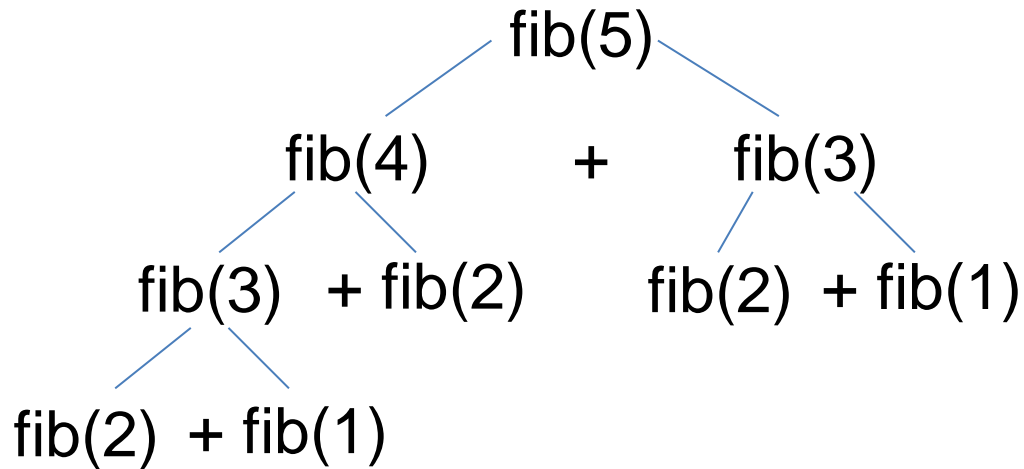
```
int fibonacci(int n){  
    if ((n == 1) || (n == 2))  
        return 1;  
    else  
        return( fibonacci(n - 1) + fibonacci(n - 2));  
}
```

Estrutura de dados



Recursividade

Exemplo: árvore recursiva



**E se fosse
 $\text{fib}(100)$???**

Estrutura de dados



Recursividade

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.

Recursividade

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.

Recursividade

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.

Estrutura de dados



Recursividade – exercício

Determine o que a seguinte função recursiva em C calcula, $n \geq 0$.

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

RECURSIVIDADE – ATIVIDADE EM TRIOS

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

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

Onde:

- $x^0 = 1$
- $n \geq 0$

ENVIAR PELO MOODLE ATÉ 12/04!

DÚVIDAS?