



INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Charqueadas

EDUCAÇÃO
PÚBLICA
100%
GRATUITA

Funções Recursivas em C

Programação Estruturada

Prof. André del Mestre

Principais Conceitos

Recursao

Conceito

- Uma função que chama a si propria
 - Chama-se **Função Recursiva**
 - Caso especial de funções
- Funções recursivas
 - Tendem a necessitar de mais memoria que as tradicionais
 - Tem codigo mais enxuto e elegante
 - Largamente utilizadas em Estrutura de Dados
 - Exemplo: funções que percorrem arvores (3º semestre do TSI)

Recursao

Conceito

- Exemplo classico de recursao na matematica: **Fatorial**
 - $4! = 4 * 3!$
 - $3! = 3 * 2!$
 - $2! = 2 * 1!$
 - $1! = 1 * 0!$
 - $0! = 1$
 - $n! = n * (n - 1)!$
 - $4! = 4 * 3 * 2 * 1$

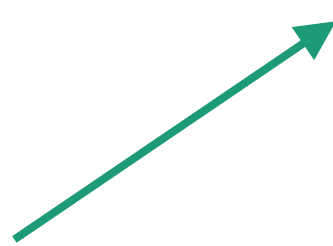
Recursao

Codificando a Funcao Fatorial em C

- Exemplo classico de recursao na matematica: **Fatorial**

- $4! = 4 * 3!$
 - $3! = 3 * 2!$
 - $2! = 2 * 1!$
 - $1! = 1 * 0!$
 - $0! = 1$
- $n! = n * (n - 1)!$
- $4! = 4 * 3 * 2 * 1$

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



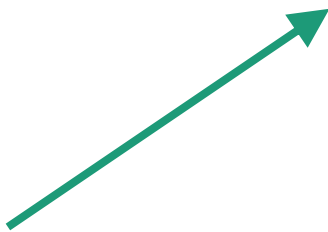
Recursao

Codificando a Funcao Fatorial em C

- Exemplo classico de recursao na matematica: **Fatorial**

- $4! = 4 * 3!$
 - $3! = 3 * 2!$
 - $2! = 2 * 1!$
 - $1! = 1 * 0!$
 - $0! = 1$
- $n! = n * (n - 1)!$
- $4! = 4 * 3 * 2 * 1$

```
int fatorial(int n) {  
    if(n<=0)  
        return 1;  
    else  
        int i, f=1;  
        for(i=n; i>0; i--) {  
            f *= i;  
        }  
    return f;  
}
```



Recursao

Comparação de função com e sem recursao

- Fatorial COM recursao

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

- Fatorial SEM recursao

```
int fatorial(int n) {  
    if(n<=0)  
        return 1;  
    else  
        int i, f=1;  
        for(i=n; i>0; i--) {  
            f *= i;  
        }  
        return f;  
}
```

Recursao

Como desenvolver Funções Recursivas?

- **Criterio de parada:** Quando a função para de chamar a si mesma

- $4! = 4 * 3!$
 - $3! = 3 * 2!$
 - $2! = 2 * 1!$
 - $1! = 1 * 0!$
 - $0! = 1$
- $n! = n * (n - 1)!$

Sabe-se que $0! = 1$,
portanto este é o
criterio de parada

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


Recursao

Como desenvolver Funções Recursivas?

- **Modificador:** parametro da função recursiva que sera **modificado** para convergir na **condição de parada**

- $4! = 4 * 3!$
 - $3! = 3 * 2!$
 - $2! = 2 * 1!$
 - $1! = 1 * 0!$
 - $0! = 1$

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

Cada nova chamada de `fatorial()`, o parametro `n` eh modificado.

Precisa convergir na direção de parada

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

Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n){
    if(n<=0)
        return 1;
    else
        int i, f=1;
        for(i=n; i>0; i--){
            f *= i;
        }
        return f;
}

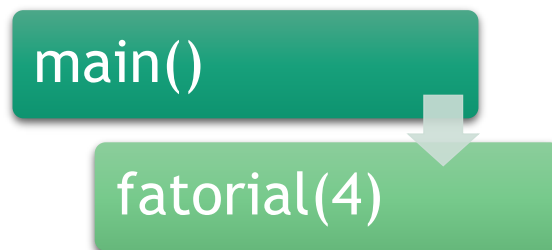
int main(){
    int res=fatorial(4);
    printf("4!=%i\n", res);
    return 0;
}
```



Funções

Gerenciamento de memória em funções recursivas

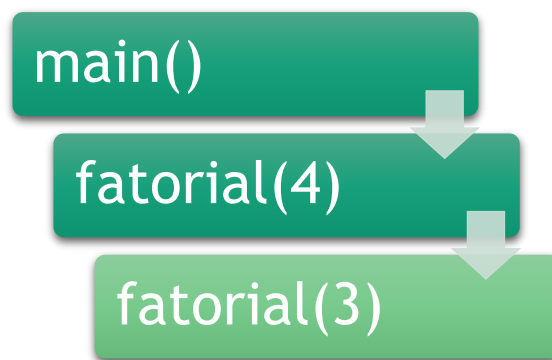
```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```



Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```

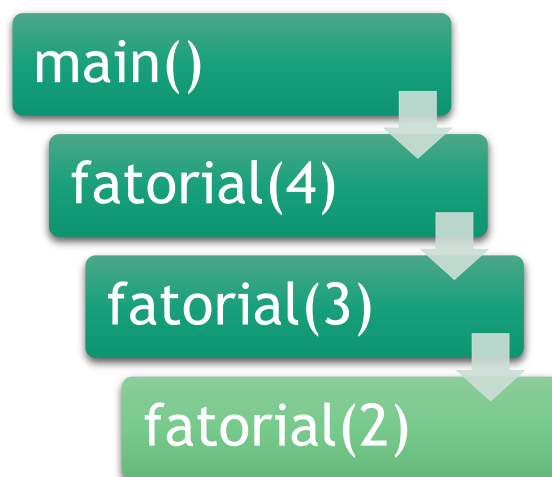


Início do empilhamento na memória: fatorial(4) não encerrou e chamou fatorial (3)

Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```



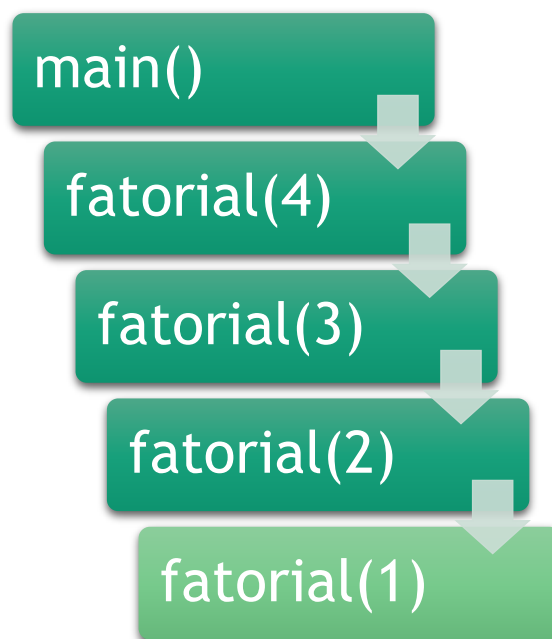
fatorial(3) não encerrou e
chamou fatorial(2)

Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```

Neste momento ha 4 instancias da função fatorial na memoria e nenhuma encerrou!

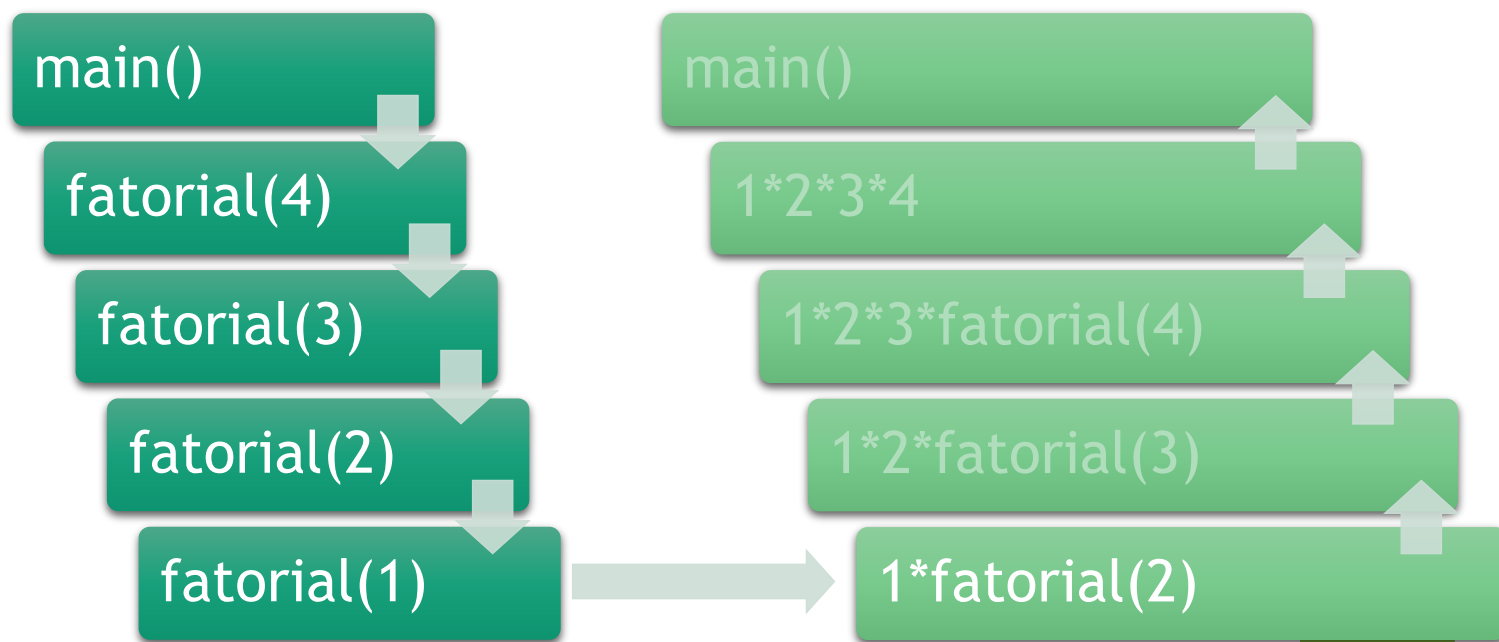


Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```

Uma instancia convergiu ao
critério de parada! Inicia o
desempilhamento da memória

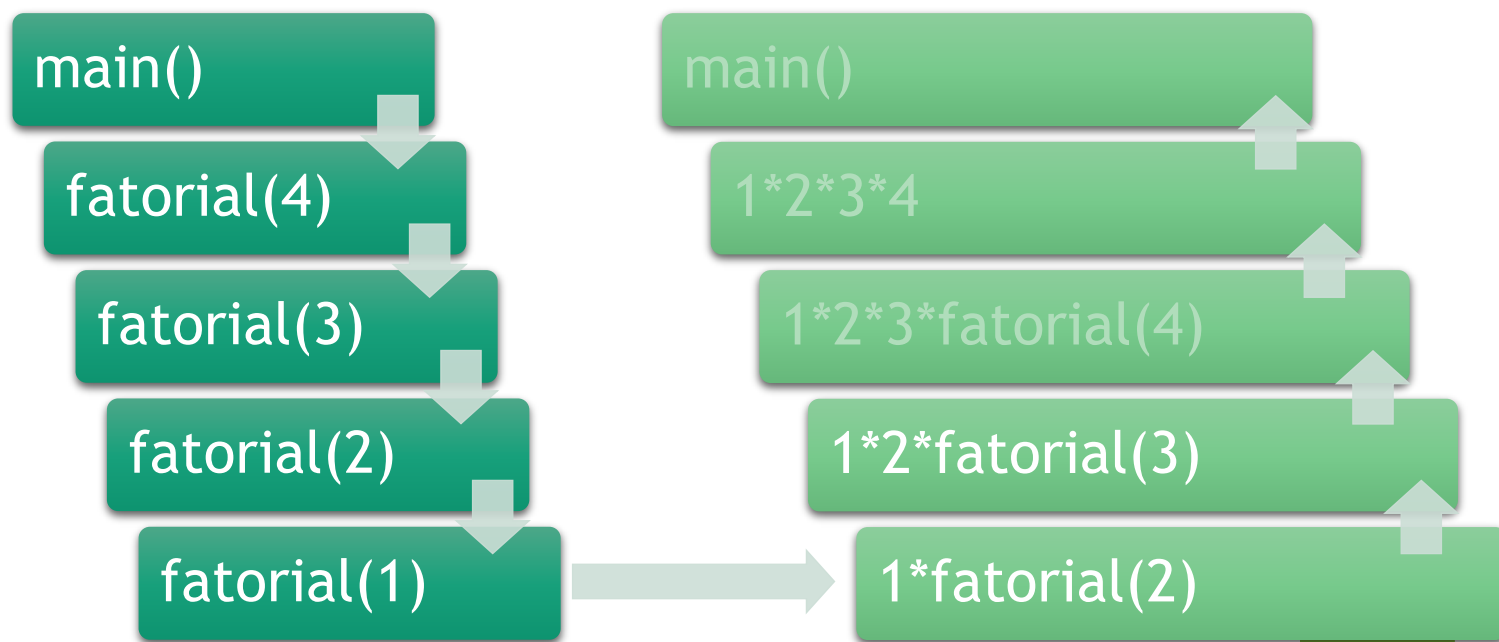


Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```

Desempilha fatorial(2)

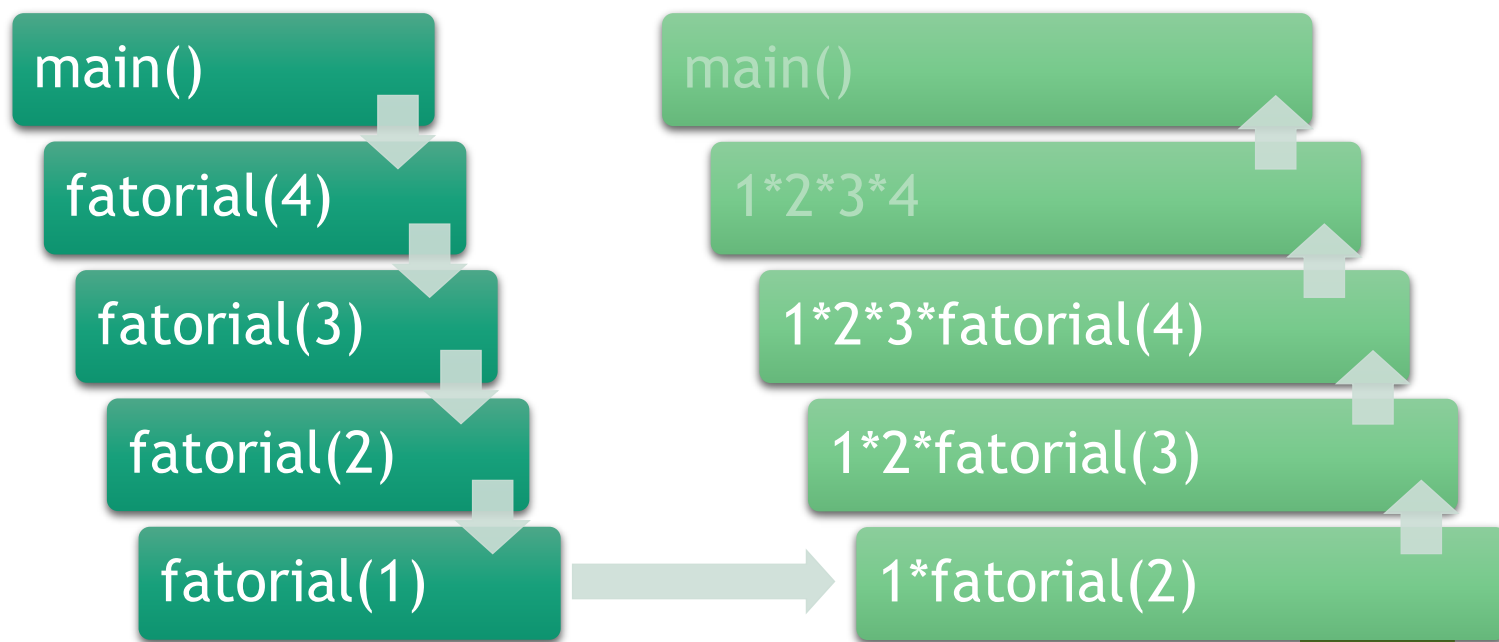


Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```

Desempilha fatorial(3)

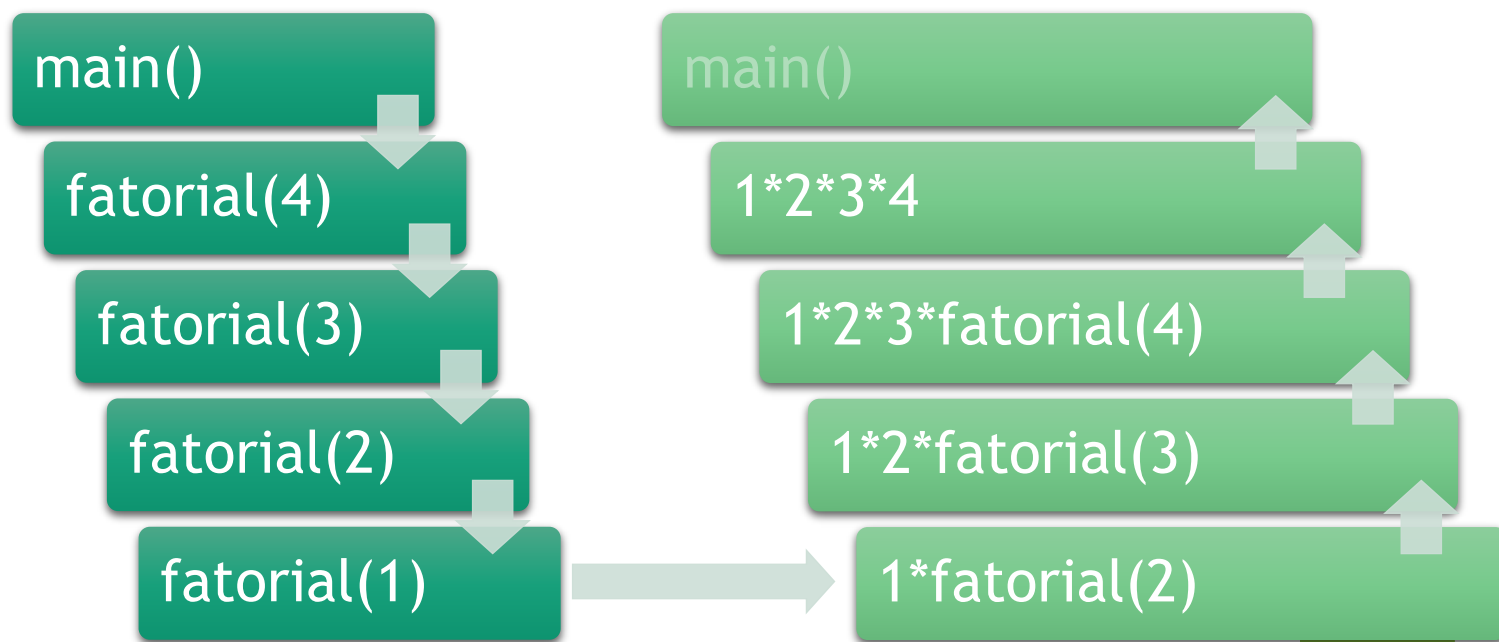


Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n) {  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main() {  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```

Desempilha fatorial(4)

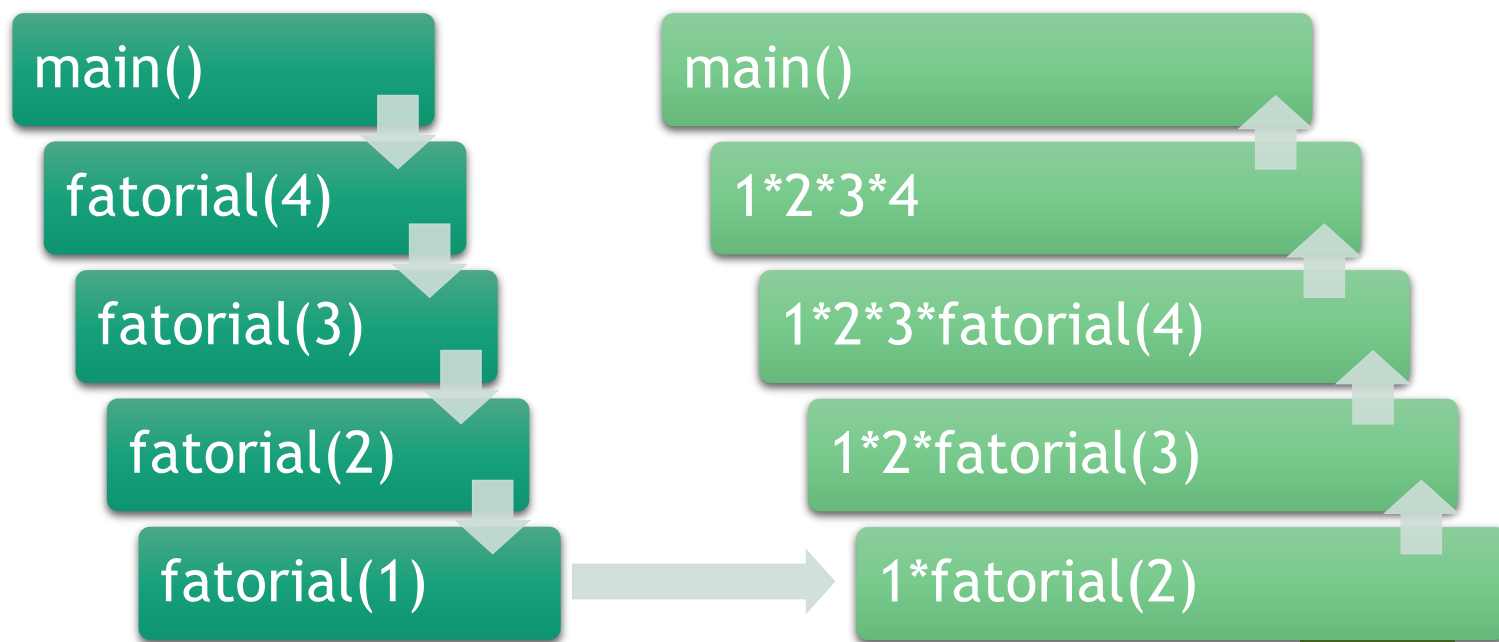


Funções

Gerenciamento de memória em funções recursivas

```
int fatorial(int n){  
    if(n<=1)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}  
int main(){  
    int res=fatorial(4);  
    printf("4!=%i\n", res);  
    return 0;  
}
```

Todas as instancias de fatorial() finalizaram e retornaram o resultado para main()



Exemplos

Recurso - Exemplo 1

Multiplicação

- Estudo de Caso: $5 * 3 = 15$

Recurção - Exemplo 1

Multiplicação

- Estudo de Caso: $\boxed{5} * \boxed{3} = 15$
A B

- $15 = \underbrace{5+5+5}$

Repetiu 3x

- $15 = \underbrace{3+3+3+3+3}$

Repetiu 5x

Recurção - Exemplo 1

Multiplicação

- Estudo de Caso: $\boxed{5} * \boxed{3} = 15$
A B

- $15 = \underbrace{5+5+5}$

Repetiu 3x

- $15 = \underbrace{3+3+3+3+3}$

Repetiu 5x

- Assumindo **B** como **modificador**

- $5*3 = 15$

- $5*3 = 5 + 5*2$

- $5*3 = 5 + 5 + 5*1$

- $5*3 = 5 + 5 + 5 + 5*0$

- $A*B = A + A*(B-1)$

- **Condição de parada**

- $B=0$

Recurção - Exemplo 1

Multiplicação

```
int multiplica(int a, int b){
    if(b>0){
        return a + multiplica(a, b-1);
    }else{
        return 0;
    }
}

int main(){
    int res= multiplica(5, 3);
    printf("5 * 3 = %d\n", res);
    return 0;
}
```

- Assumindo **B** como **modificador**

- $5*3 = 15$
 - $5*3 = 5 + 5*2$
 - $5*3 = 5 + 5 + 5*1$
 - $5*3 = 5 + 5 + 5 + 5*0$
- $A*B = A + A*(B-1)$

- Condição de parada**

- $B=0$

Recurção - Exemplo 2

Divisão

- Estudo de Caso 1: $\boxed{15} / \boxed{3} = 5$
A B

- $15/3 = 5$
 - $15/3 = 1 + 12/3$
 - $15/3 = 1 + 1 + 9/3$
 - $15/3 = 1 + 1 + 1 + 6/3$
 - $15/3 = 1 + 1 + 1 + 1 + 3/3$
 - $15/3 = 1 + 1 + 1 + 1 + 1 + 0/3$
- $A/B = 1 + (A-B)/B$

Recurção - Exemplo 2

Divisão

• Estudo de Caso 1: $\boxed{15} / \boxed{3} = 5$

A B

- $15/3 = 5$
 - $15/3 = 1 + 12/3$
 - $15/3 = 1 + 1 + 9/3$
 - $15/3 = 1 + 1 + 1 + 6/3$
 - $15/3 = 1 + 1 + 1 + 1 + 3/3$
 - $15/3 = 1 + 1 + 1 + 1 + 1 + 0/3$
- $A/B = 1 + (A-B)/B$

• Estudo de Caso 2: $\boxed{17} / \boxed{3} = 5$

A B

- $17/3 = 5$
 - $17/3 = 1 + 14/3$
 - $17/3 = 1 + 1 + 11/3$
 - $17/3 = 1 + 1 + 1 + 8/3$
 - $17/3 = 1 + 1 + 1 + 1 + 5/3$
 - $17/3 = 1 + 1 + 1 + 1 + 1 + \boxed{2/3}$
- $A/B = 1 + (A-B)/B$

Recurção - Exemplo 2

Divisão

- Parametro **A** eh o **modificador**

- $17/3 = 5$

- $17/3 = 1 + 14/3$
- $17/3 = 1 + 1 + 11/3$
- $17/3 = 1 + 1 + 1 + 8/3$
- $17/3 = 1 + 1 + 1 + 1 + 5/3$
- $17/3 = 1 + 1 + 1 + 1 + 1 + 2/3$

- $A/B = 1 + (A-B)/B$

- Estudo de Caso 3: $\boxed{2} / \boxed{3} = 5$

2

↓

A

3

↓

B

- $2/3 = 0$
 - $2/3 = 0$

- Assim,
 - $A/B = 1 + (A-B)/B$, se $A \geq B$
 - $A/B = 0$, se $A < B$

- Logo, **condição de parada**
 - $A < B$

Recurção - Exemplo 2

Divisão

```
// Versao 1
// Utilizando pilha da memoria
int divisaoV1(int a, int b){
    if(a>=b){
        return 1+divisaoV1(a-b, b);
    }else{
        return 0;
    }
}

int main(){
    int res=divisaoV1(15, 3);
    printf("15 / 3 = %d\n", res);
    return 0;
}
```

```
// Versao 2
// Utilizando var auxiliar - ctrl
int divisaoV2(int a, int b, int ctrl){
    if(a>=b){
        return divisaoV2(a-b, b, ctrl+1);
    }else{
        return ctrl;
    }
}

int main(){
    int res=divisaoV2(15, 3, 0);
    printf("15 / 3 = %d\n", res);
    return 0;
}
```

MUITO
OBRIGADO

Prof. André del Mestre

www.ifsul.edu.br
almmartins@charqueadas.ifsul.edu.br