

Recursividade

Algoritmos e Estruturas de Dados I

Prof. Lucas Astore

Prof. Cristiano Rodrigues



Função recursiva

Uma função é dita recursiva quando ela **chama a si mesma**.

Uma função recursiva normalmente apresenta duas características básicas:

1. Chamada recursiva
2. Condição de parada

Tinham dois cachorros, o Pete
e o Repete. O Pete morreu,
quem ficou?

Qual a condição de
parada da história?

Exemplo 01

Exemplo – procedimento iterativo

Escreva um procedimento **iterativo** chamado **mostrar()** que mostre na tela os números de 0 a 3.

Utilize a estrutura de repetição **for**.

Exemplo – procedimento recursivo

Escreva um procedimento **recursivo** chamado **mostrar()** que mostre na tela os números 0 a 3.

Utilize o conceito de recursão para resolver o problema, sem usar estruturas de repetição como `for` ou `while`.

Exemplo de Procedimento Iterativo vs Recursivo


- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

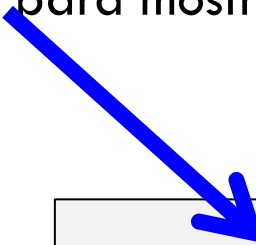
```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
    Else return  
}
```

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3



```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```



```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

Tela
1
2
3
4

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

Tela
1
2
3
4

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

i	0
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

true

Tela
1
2
3
4

i	0
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {
    for (int i = 0; i < 4; i = i + 1) {
        printf("%d", i);
    }
}
```

```
void mostrar () {
    mostrar (0);
}
void mostrar (int i) {
    if (i < 4) {
        printf("%d", i);
        mostrar (i + 1);
    }
}
```

Tela
0
2
3
4

i	0
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

Tela
0
2
3
4

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

i	1
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

Tela
0
2
3
4

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

i	1
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

true

Tela
0

i	1
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

Tela
0
1

i	1
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {
    for (int i = 0; i < 4; i = i + 1) {
        printf("%d", i);
    }
}
```

Tela
0
1

```
void mostrar () {
    mostrar (0);
}

void mostrar (int i) {
    if (i < 4) {
        printf("%d", i);
        mostrar (i + 1);
    }
}
```

i	2
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

Tela
0
1

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

i	2
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

true

Tela
0
1

i	2
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

Tela
0
1
2

i	2
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

Tela
0
1
2

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

i	3
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

Tela
0
1
2

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

i	3
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

true

Tela
0
1
2

i	3
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

Tela
0
1
2
3

i	3
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {
    for (int i = 0; i < 4; i = i + 1) {
        printf("%d", i);
    }
}
```

Tela
0
1
2
3

```
void mostrar () {
    mostrar (0);
}

void mostrar (int i) {
    if (i < 4) {
        printf("%d", i);
        mostrar (i + 1);
    }
}
```

i	4
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

Tela
0
1
2
3

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

i	4
---	---

Exemplo de Procedimento Iterativo vs Recursivo

- Procedimentos ITERATIVO e RECURSIVO para mostrar os números 0 à 3

```
void mostrar () {  
    for (int i = 0; i < 4; i = i + 1) {  
        printf("%d", i);  
    }  
}
```

```
void mostrar () {  
    mostrar (0);  
}  
void mostrar (int i) {  
    if (i < 4) {  
        printf("%d", i);  
        mostrar (i + 1);  
    }  
}
```

false

Tela
0
1
2
3

i	4
---	---

Exemplo 02

Exercício

- Identifique as chamadas recursivas e condições de parada

```
int fat (int n){  
    int resp;  
    if (n == 1){  
        resp = 1;  
    } else {  
        resp = n * fat (n - 1);  
    }  
    return resp;  
}
```

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

Exercício

- Identifique as chamadas recursivas e condições de parada

```
int fat (int n){  
    int resp;  
    if (n == 1){  
        resp = 1;  
    } else {  
        resp = n * fat (n - 1);  
    }  
    return resp;  
}
```

Chamadas recursivas

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


Exercício

- Identifique as chamadas recursivas e condições de parada

```
int fat (int n){  
    int resp;  
    if (n == 1){  
        resp = 1;  
    } else {  
        resp = n * fat(n - 1);  
    }  
    return resp;  
}
```

A cada chamada recursiva,
o n se aproxima do último valor

Condições de parada

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

Exemplo 03

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1o – inicio");  
    segundo();  
    printf("1o – fim");  
}  
  
void segundo(){  
    printf("2o – inicio e fim");  
}  
  
void main (){  
    printf("main – inicio");  
    primeiro();  
    printf("main – fim");  
}
```

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1o – início");  
    segundo();  
    printf("1o – fim");  
}  
  
void segundo(){  
    printf("2o – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1o – início");  
    segundo();  
    printf("1o – fim");  
}  
  
void segundo(){  
    printf("2o – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1o – início");  
    segundo();  
    printf("1o – fim");  
}  
  
void segundo(){  
    printf("2o – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1o – início");  
    segundo();  
    printf("1o – fim");  
}  
  
void segundo(){  
    printf("2o – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

2º – início e fim

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

2º – início e fim

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

2º – início e fim

1º – fim

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

2º – início e fim

1º – fim

Exemplo de Procedimento Iterativo

- O que o programa iterativo abaixo mostra na tela?

```
void primeiro(){  
    printf("1º – início");  
    segundo();  
    printf("1º – fim");  
}  
  
void segundo(){  
    printf("2º – início e fim");  
}  
  
void main (){  
    printf("main – início");  
    primeiro();  
    printf("main – fim");  
}
```

TELA

main – início

1º – início

2º – início e fim

1º – fim

main – fim

Exemplo 04

Exercício

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
void printR(){
    printRecursivo(2);
}

void printRecursivo(int i){
    printf("%d",i);
    if (i > 0){
        printRecursivo(i - 1);
    }
    printf("%d",i);
}
```

Exercício

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
void printR(){
    printRecursivo(2);
}

void printRecursivo(int i){
    printf("%d",i);
    if (i > 0){
        printRecursivo(i - 1);
    }
    printf("%d",i);
}
```

Temos como se cada chamada recursiva fosse uma função diferente!!!

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
void printR(){  
    printRecursivo(2);  
}
```

```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
(1) void printR(){  
(2)     printRecursivo(2);  
}
```

```
(3) void printRecursivo(int i){ // i (2)  
(4)     printf("%d",i);  
(5)     if (i > 0){  
(6)         printRecursivo(i - 1);  
        }  
        printf("%d",i);  
}
```

```
(7) void printRecursivo(int i){ // i (1)  
(8)     printf("%d",i);  
(9)     if (i > 0){  
(10)        printRecursivo(i - 1);  
        }  
        printf("%d",i);  
}
```

```
(11) void printRecursivo(int i){ // i (0)  
(12)     printf("%d",i);  
(13)     if (i > 0){  
        printRecursivo(i - 1);  
        }  
(14)     printf("%d",i);  
}
```

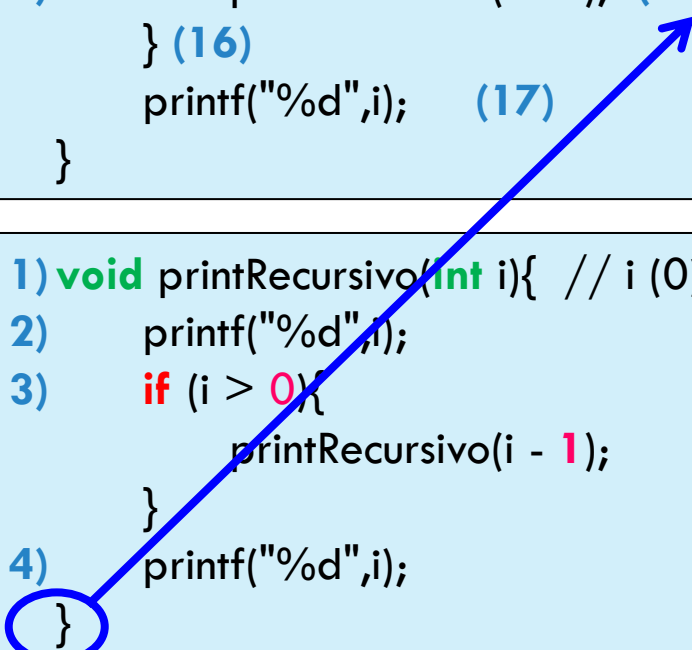
Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
(1) void printR(){  
(2)     printRecursivo(2);  
}
```

```
(3) void printRecursivo(int i){ // i (2)  
(4)     printf("%d",i);  
(5)     if (i > 0){  
(6)         printRecursivo(i - 1);  
        }  
        printf("%d",i);  
}
```

```
(7) void printRecursivo(int i){ // i (1)  
(8)     printf("%d",i);  
(9)     if (i > 0){  
(10)        printRecursivo(i - 1); (15)  
        } (16)  
        printf("%d",i); (17)  
}
```



```
(11) void printRecursivo(int i){ // i (0)  
(12)     printf("%d",i);  
(13)     if (i > 0){  
        printRecursivo(i - 1);  
    }  
(14)     printf("%d",i);  
    }
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
(1) void printR(){  
(2)     printRecursivo(2);  
}
```

```
(3) void printRecursivo(int i){ // i (2)  
(4)     printf("%d",i);  
(5)     if (i > 0){  
(6)         printRecursivo(i - 1); (18)  
            } (19)  
            printf("%d",i); (20)  
}
```

```
(7) void printRecursivo(int i){ // i (1)  
(8)     printf("%d",i);  
(9)     if (i > 0){  
(10)        printRecursivo(i - 1); (15)  
            } (16)  
            printf("%d",i); (17)  
            }
```

```
(11) void printRecursivo(int i){ // i (0)  
(12)     printf("%d",i);  
(13)     if (i > 0){  
            printRecursivo(i - 1);  
        }  
(14)     printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
(1) void printR(){  
(2)     printRecursivo(2); (21)  
}
```



```
(3) void printRecursivo(int i){ // i (2)  
(4)     printf("%d",i);  
(5)     if (i > 0){  
(6)         printRecursivo(i - 1); (18)  
        } (19)  
        printf("%d",i); (20)  
    }
```

```
(7) void printRecursivo(int i){ // i (1)  
(8)     printf("%d",i);  
(9)     if (i > 0){  
(10)         printRecursivo(i - 1); (15)  
        } (16)  
        printf("%d",i); (17)  
    }
```

```
(11) void printRecursivo(int i){ // i (0)  
(12)     printf("%d",i);  
(13)     if (i > 0){  
        printRecursivo(i - 1);  
    }  
(14)     printf("%d",i);  
    }
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
(1) void printR(){  
(2)     printRecursivo(2); (21)  
    }
```

```
(3) void printRecursivo(int i){ // i (2)  
(4)     printf("%d",i);  
(5)     if (i > 0){  
(6)         printRecursivo(i - 1); (18)  
    } (19)  
    printf("%d",i); (20)  
}
```

```
(7) void printRecursivo(int i){ // i (1)  
(8)     printf("%d",i);  
(9)     if (i > 0){  
(10)        printRecursivo(i - 1); (15)  
    } (16)  
    printf("%d",i); (17)  
}
```

```
(11) void printRecursivo(int i){ // i (0)  
(12)     printf("%d",i);  
(13)     if (i > 0){  
        printRecursivo(i - 1);  
    }  
(14)     printf("%d",i);  
}
```


Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2? ... Vamos de novo...

```
void printR(){  
    printRecursivo(2);  
}
```

```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
void printR(){  
    printRecursivo(2);  
}
```

```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```


```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
void printR(){  
    printRecursivo(2);  
}
```



```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```


```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
void printR(){  
    printRecursivo(2);  
}
```



```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```


```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime **2**, 1, 0, 0, 1 e 2?

```
void printR(){  
    printRecursivo(2);  
}
```



```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

true


```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?

```
void printR(){  
    printRecursivo(2);  
}
```



```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1); (a)  
    }  
    printf("%d",i);  
}
```

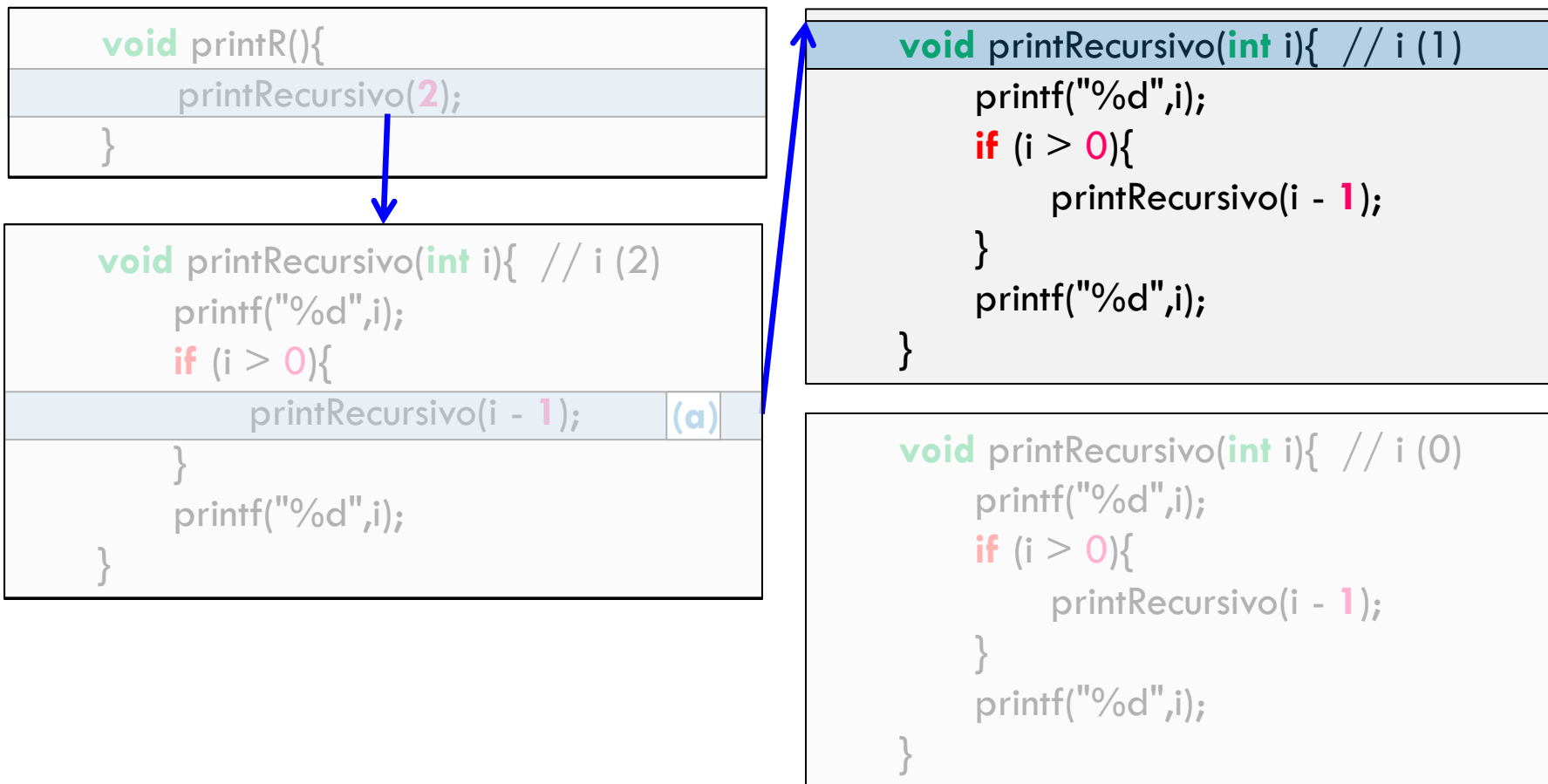
Vamos para o print do um,
contudo, depois, voltaremos para (a)

```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

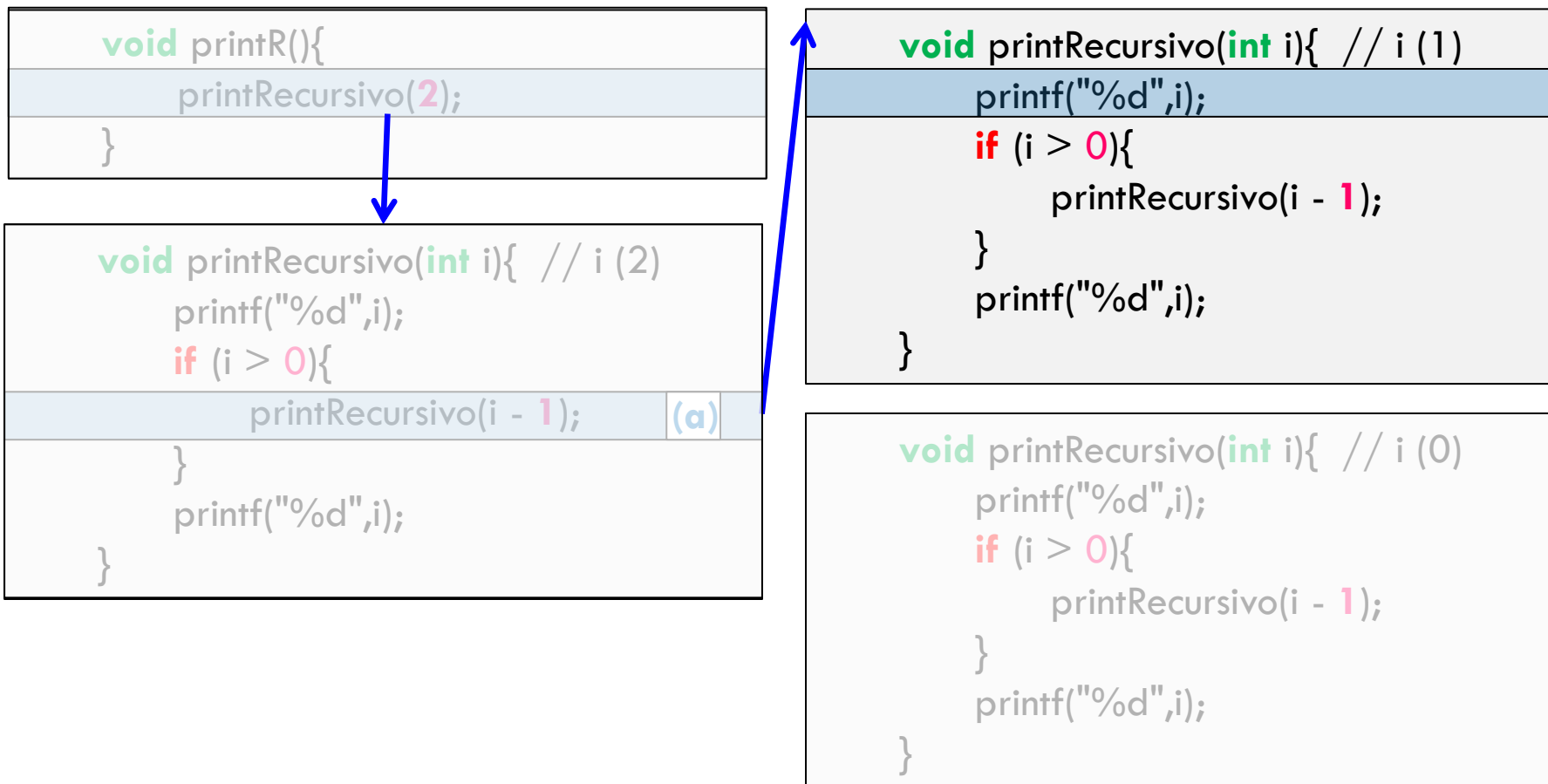
Exercício - Reavaliando

- Por que o código abaixo imprime 2, 1, 0, 0, 1 e 2?



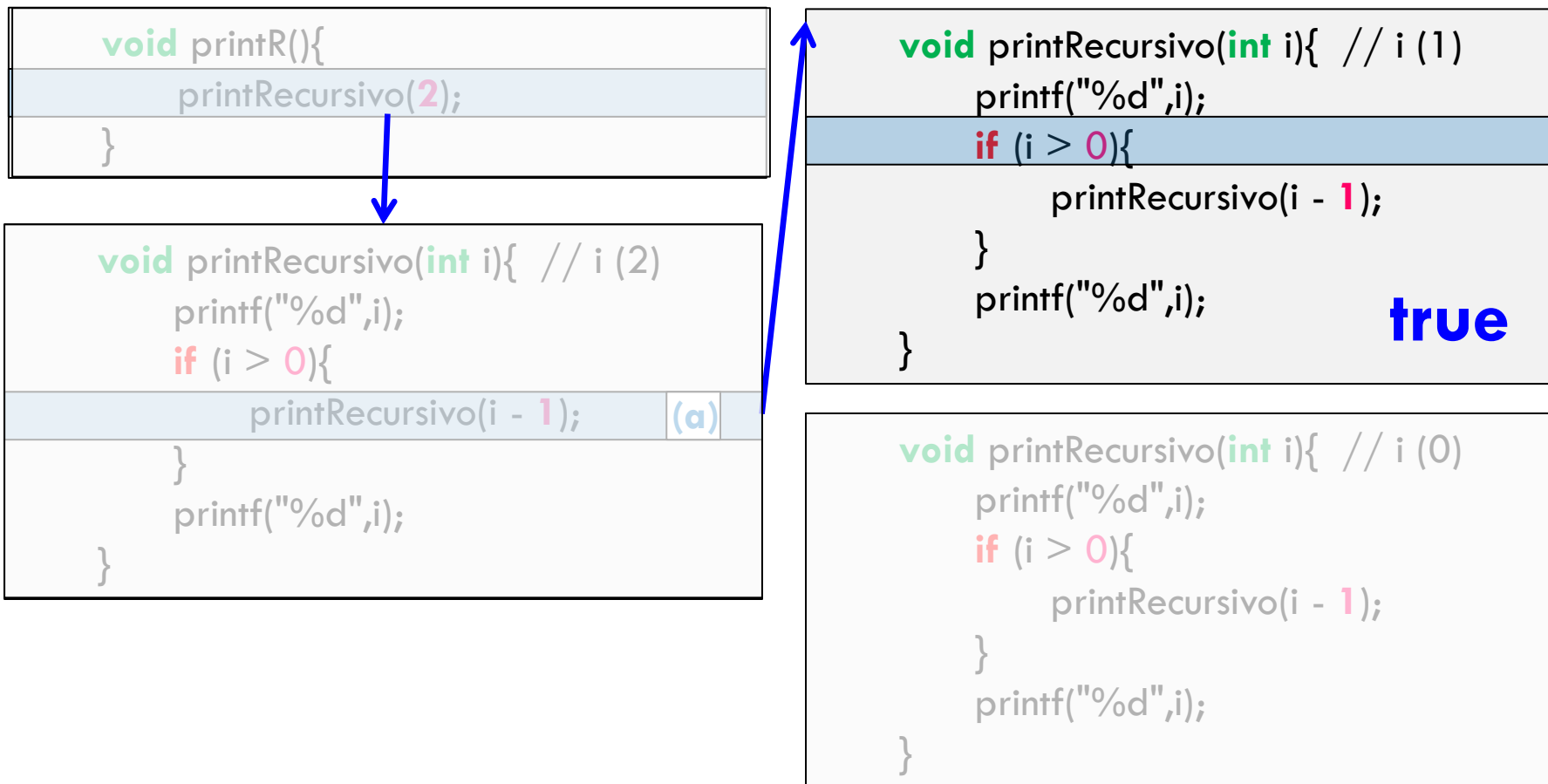
Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, 0, 0, 1 e 2?



Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, 0, 0, 1 e 2?



Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, 0, 0, 1 e 2?

```
void printR(){  
    printRecursivo(2);  
}
```

```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1); (a)  
    }  
    printf("%d",i);  
}
```

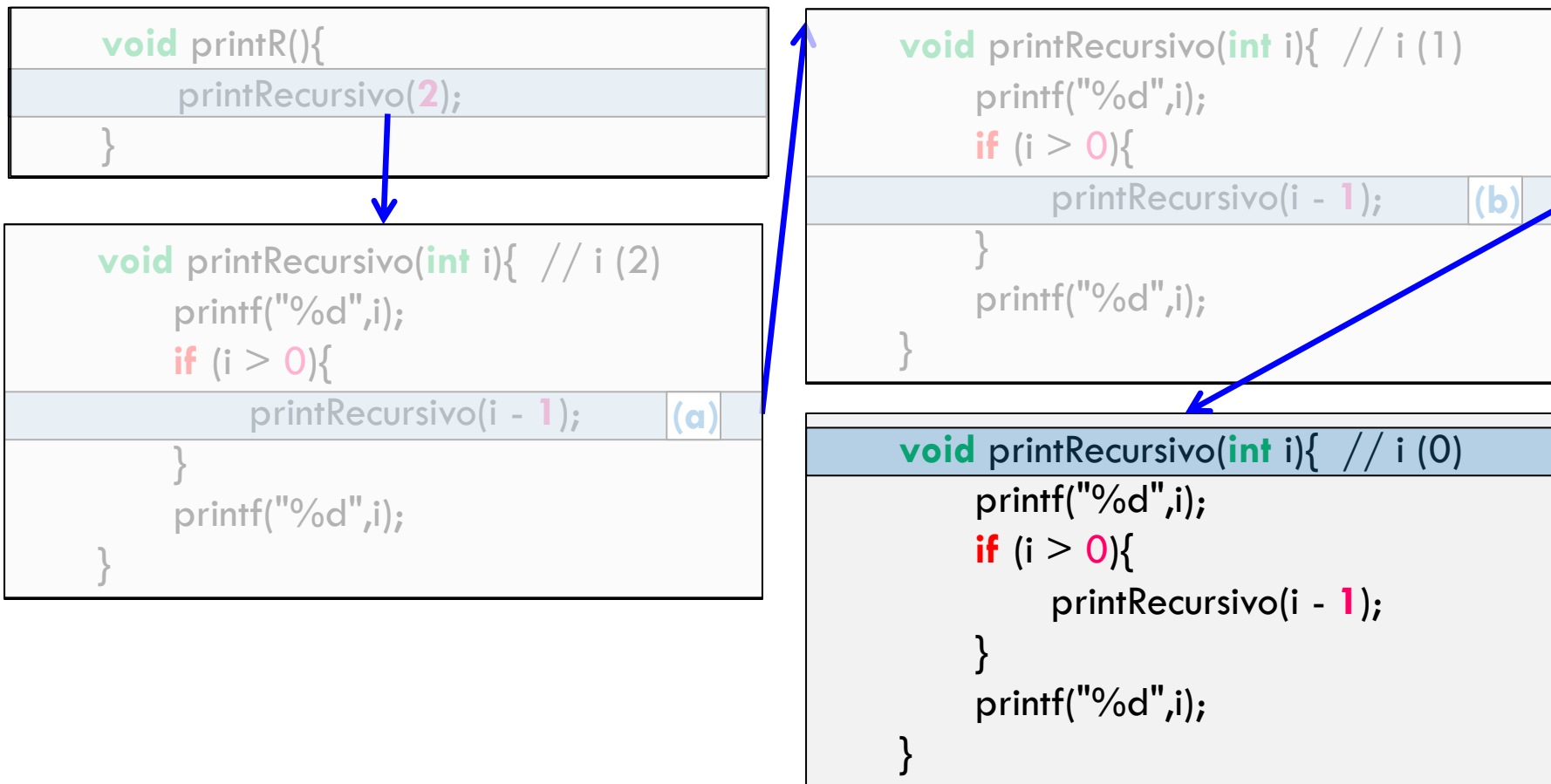
Vamos para o print do zero,
contudo, depois, voltaremos para (b)

```
void printRecursivo(int i){ // i (1)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1); (b)  
    }  
    printf("%d",i);  
}
```

```
void printRecursivo(int i){ // i (0)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

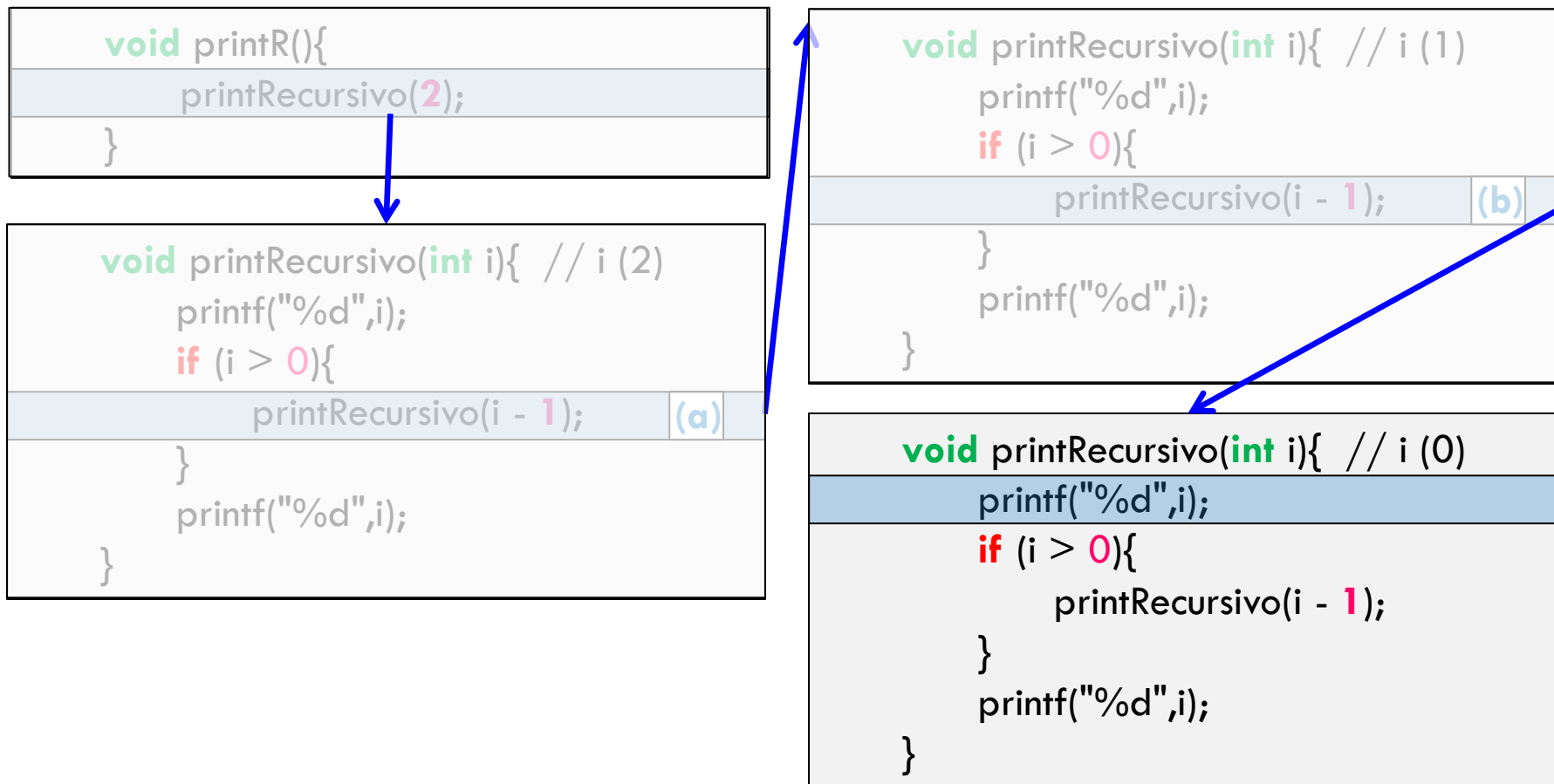
Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, 0, 0, 1 e 2?



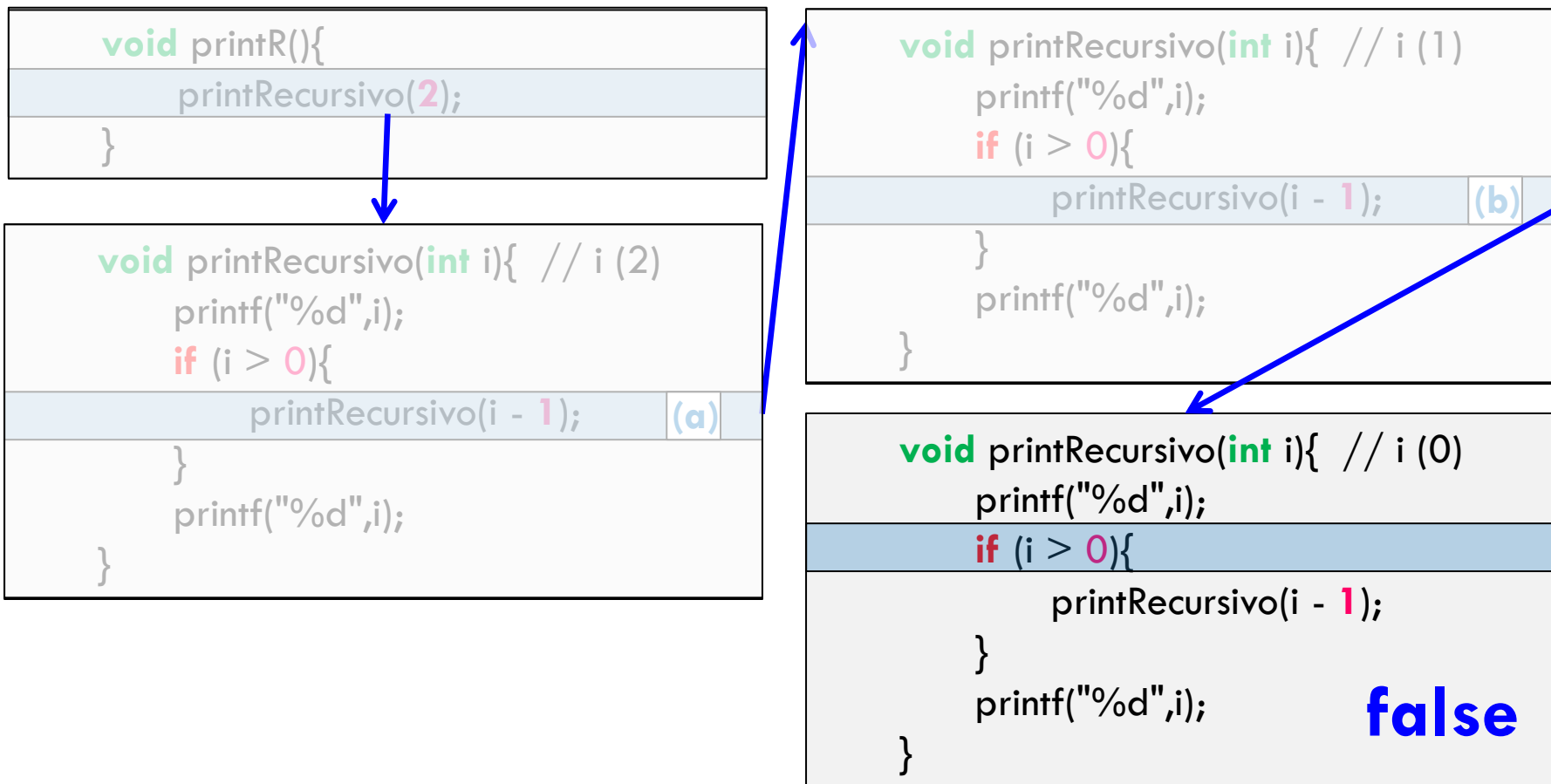
Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, **0**, 0, 1 e 2?



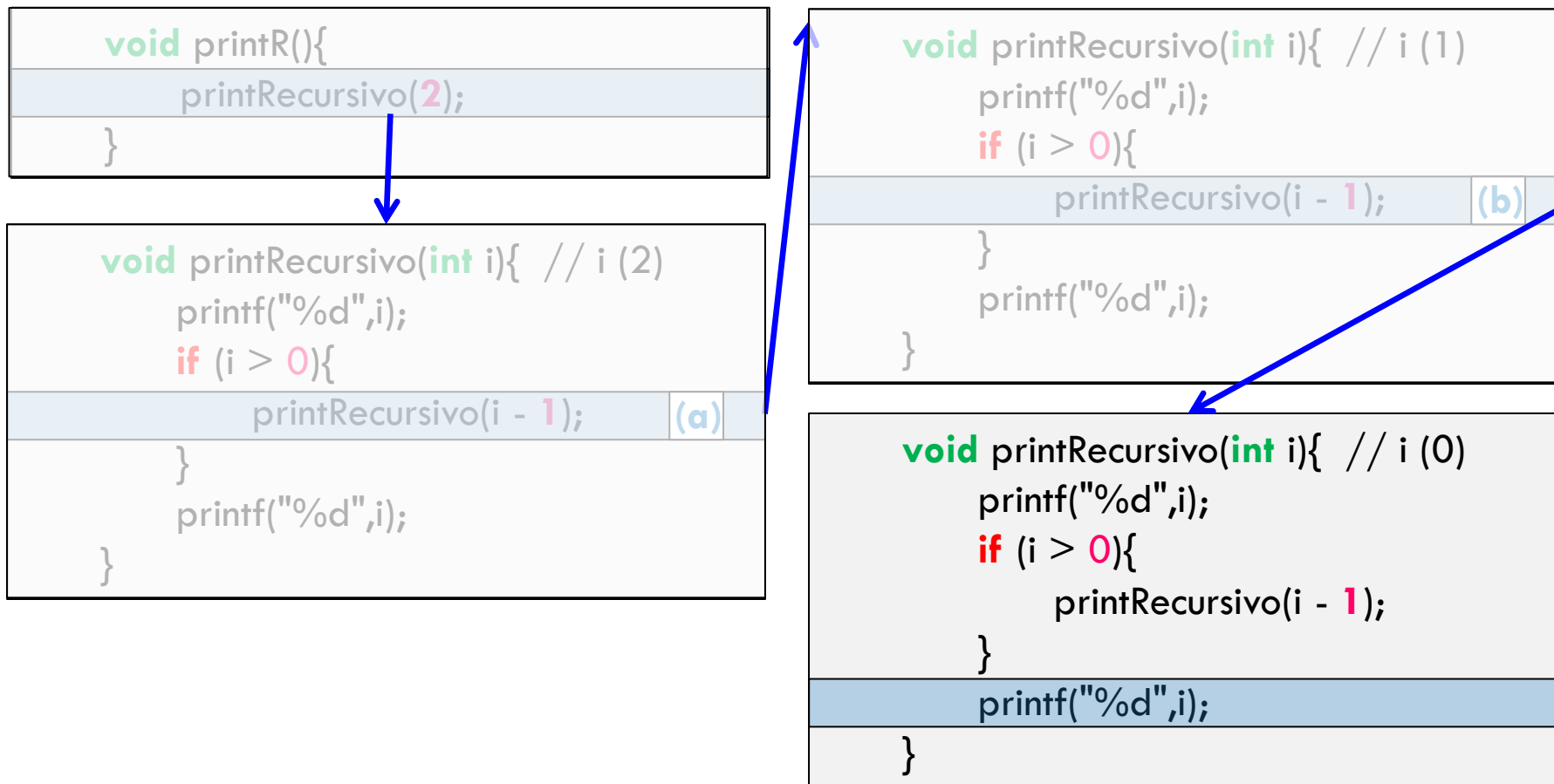
Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?



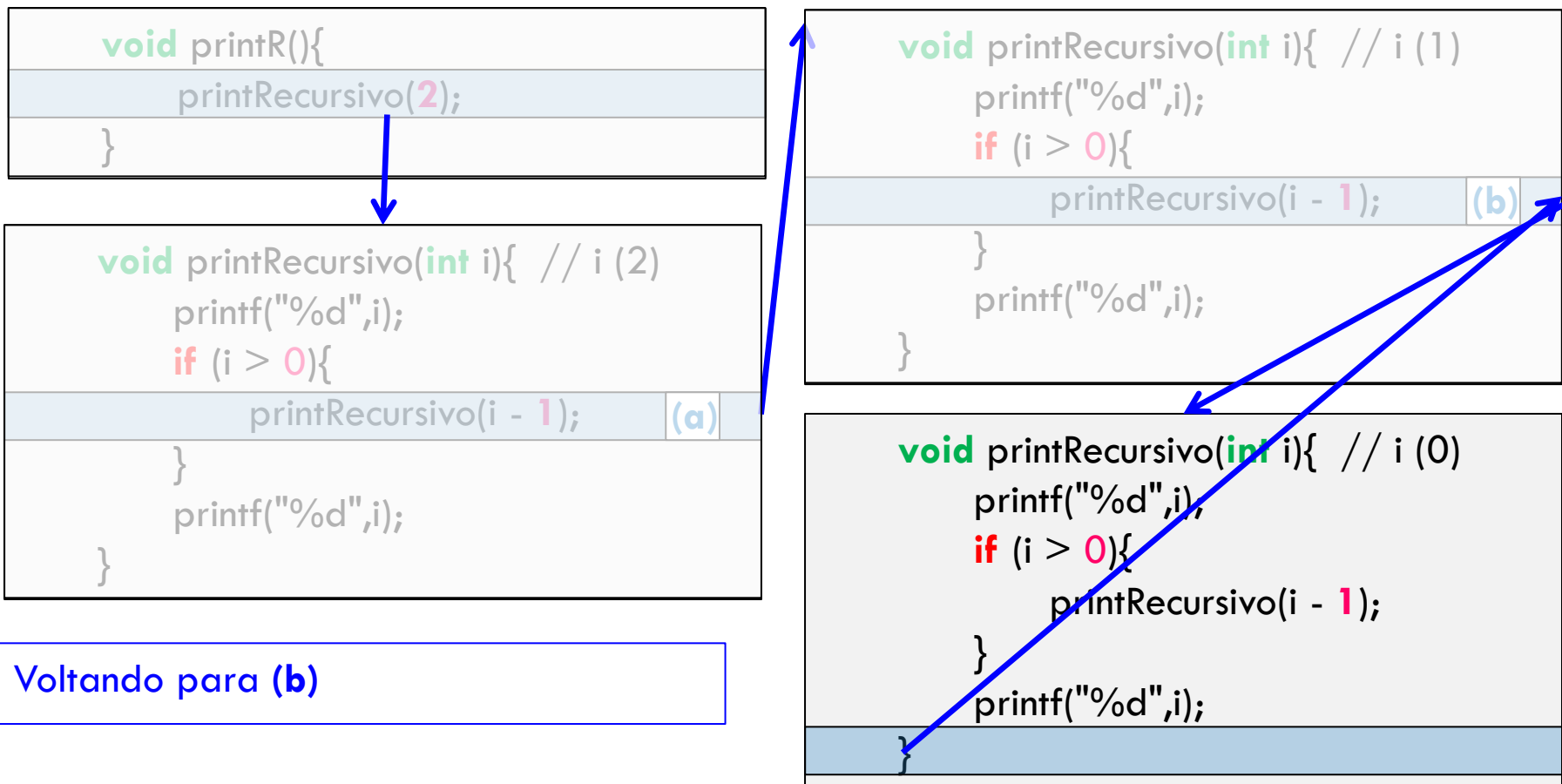
Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?



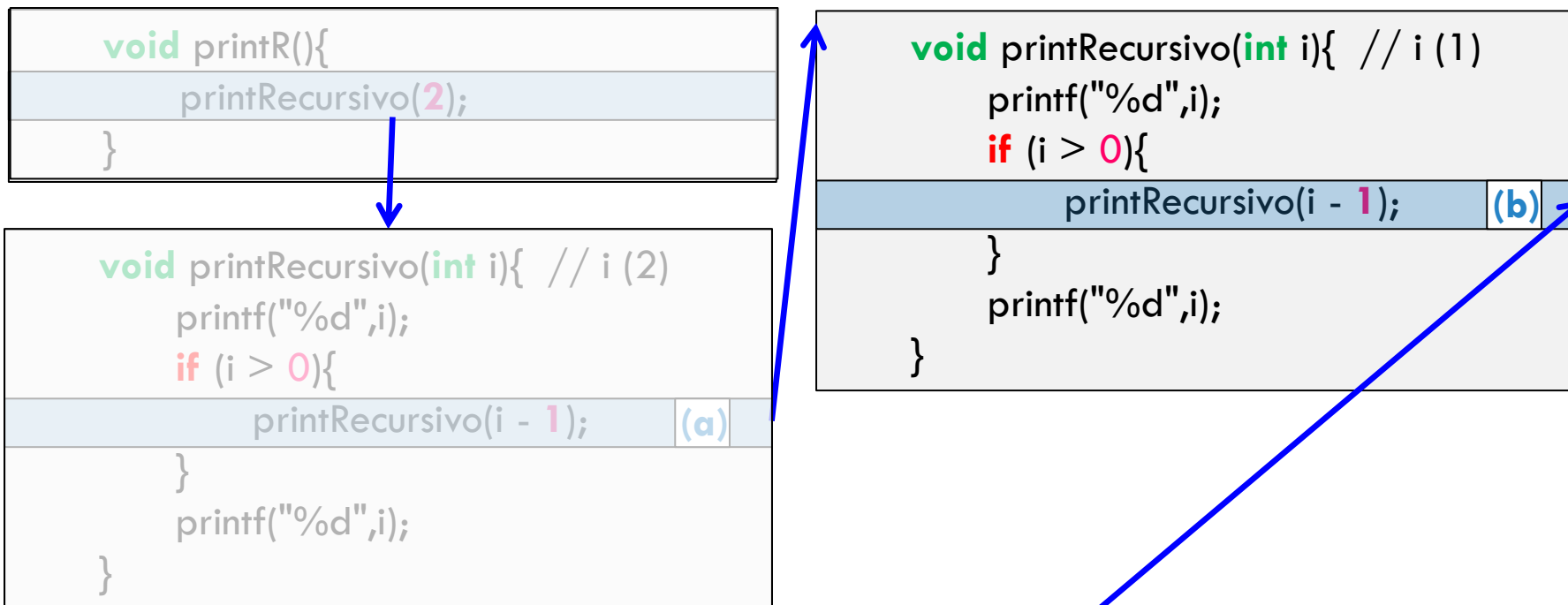
Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?



Exercício - Reavaliando

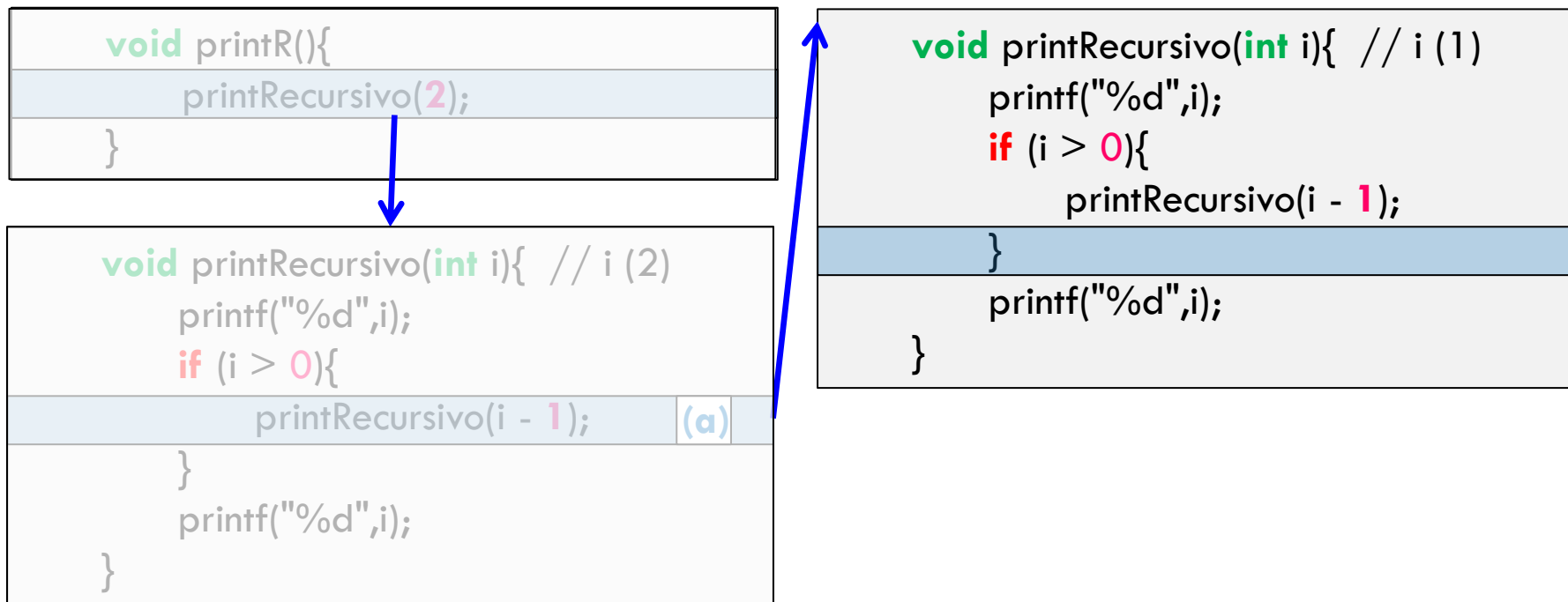
- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?



Voltando para (b)

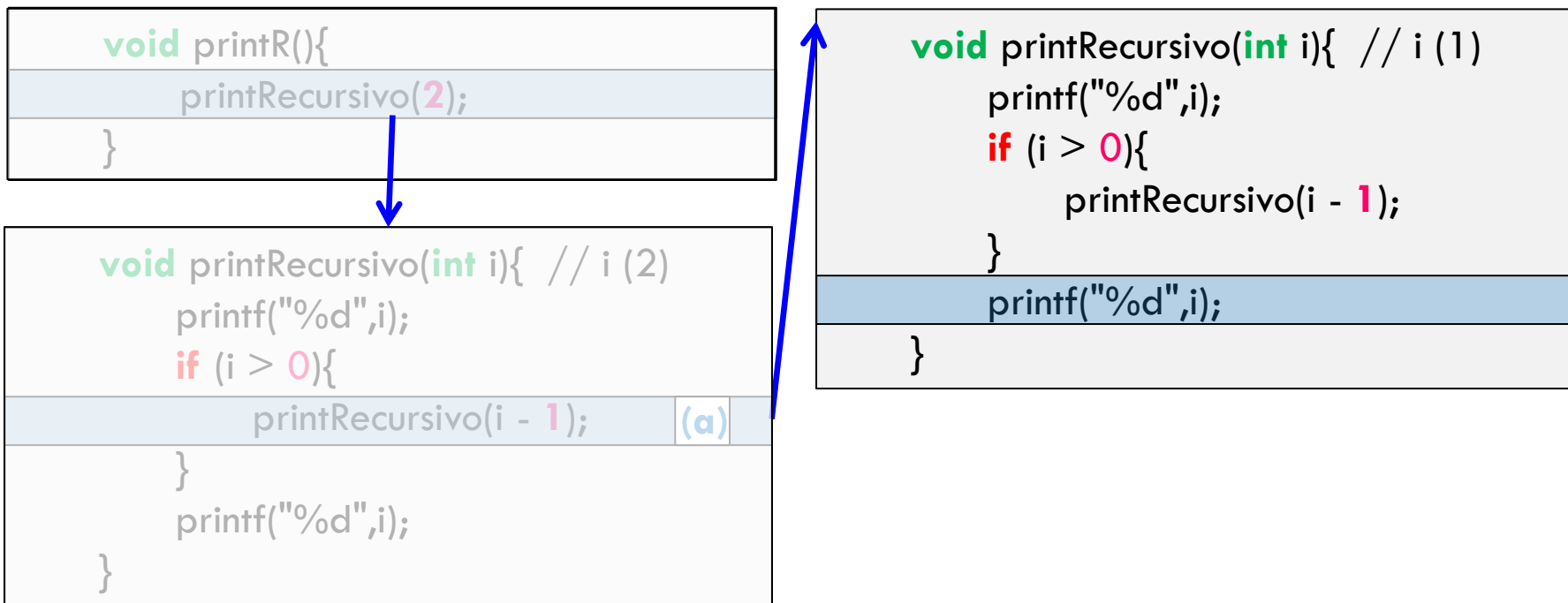
Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, **0**, **0**, **1** e **2**?



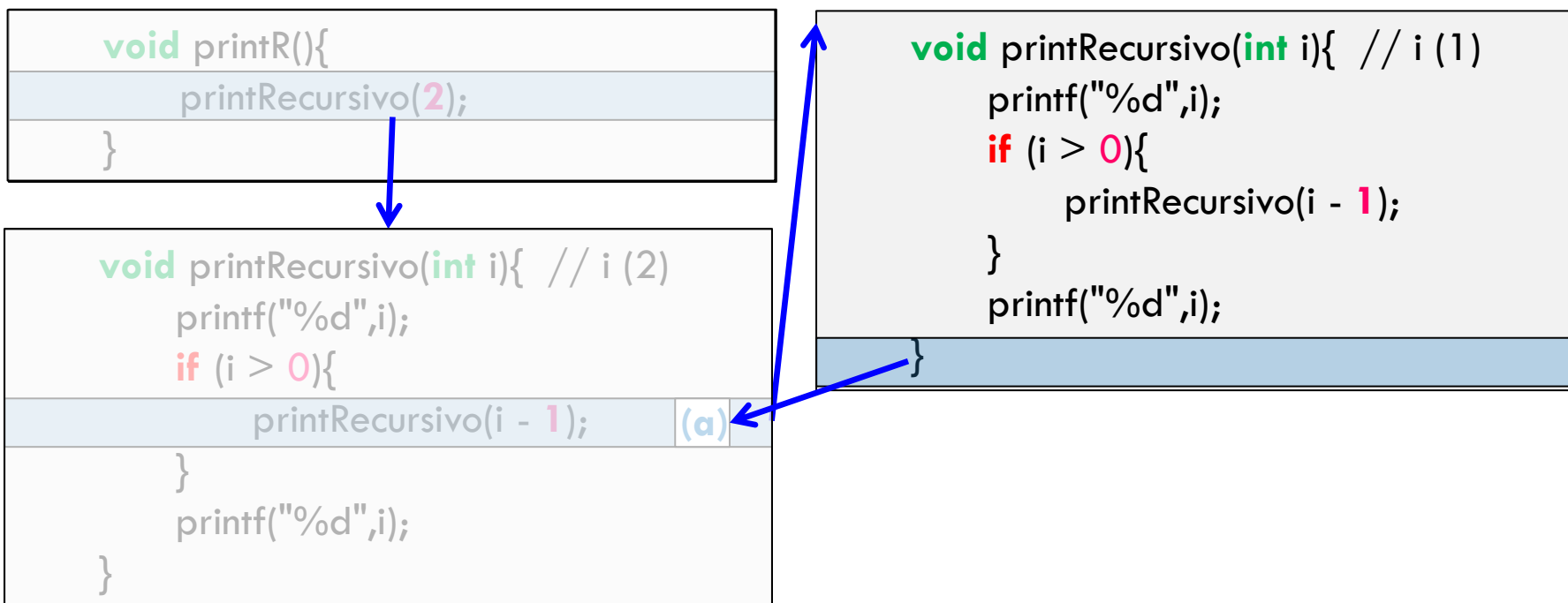
Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?



Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?



Voltando para **(a)**

Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?

```
void printR(){  
    printRecursivo(2);  
}
```

```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Voltando para (a)

Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e 2?

```
void printR(){  
    printRecursivo(2);  
}
```




```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, **0**, **0**, **1** e **2**?

```
void printR(){  
    printRecursivo(2);  
}
```



```
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Exercício - Reavaliando

- Por que o código abaixo imprime **2, 1, 0, 0, 1** e **2**?

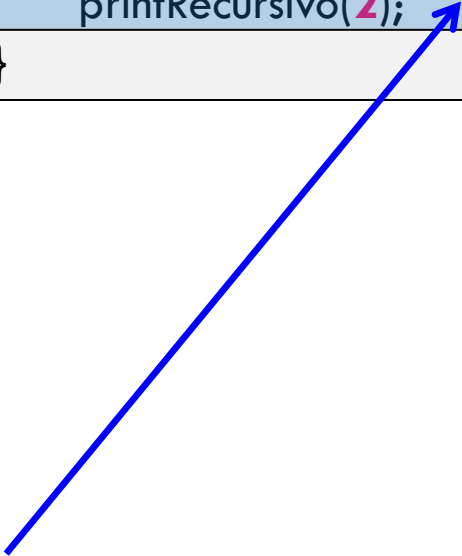
```
void printR(){  
    printRecursivo(2);  
}  
  
void printRecursivo(int i){ // i (2)  
    printf("%d",i);  
    if (i > 0){  
        printRecursivo(i - 1);  
    }  
    printf("%d",i);  
}
```

Voltando para **(primeiro)**

Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, **0**, **0**, **1** e **2**?

```
void printR(){  
    printRecursivo(2);  
}
```



Exercício - Reavaliando

- Por que o código abaixo imprime **2**, **1**, **0**, **0**, **1** e **2**?

```
void printR(){  
    printRecursivo(2);  
}
```

Exemplo 05

Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 5 * \text{Fat}(4)$$

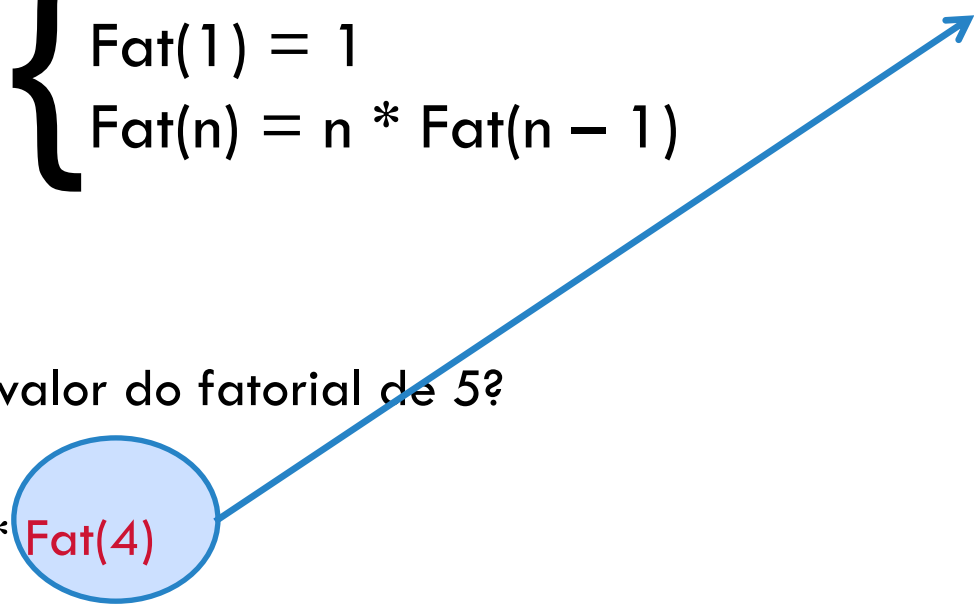
Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

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

- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 5 * \text{Fat}(4)$$


Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 5 * \text{Fat}(4)$$

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

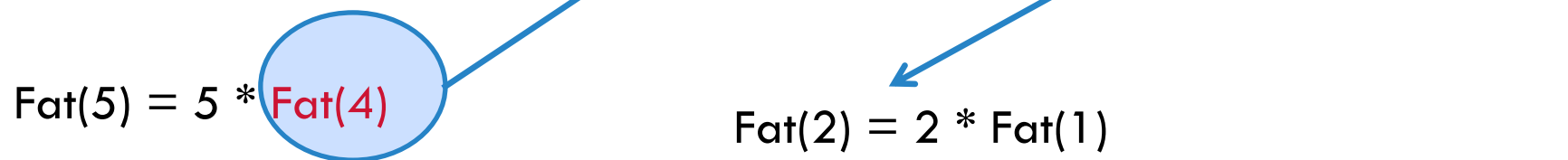
$$\text{Fat}(3) = 3 * \text{Fat}(2)$$

Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

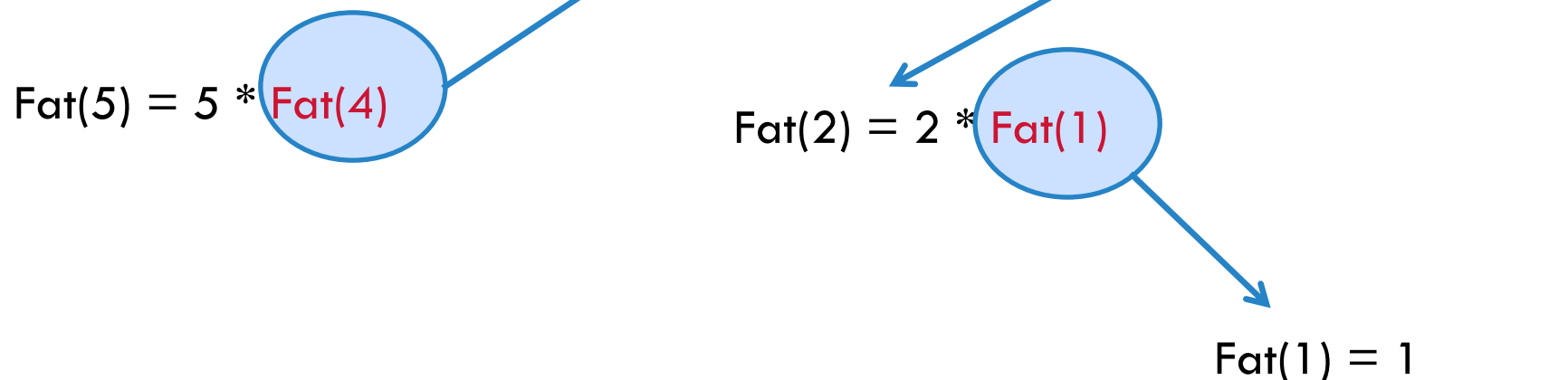


Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

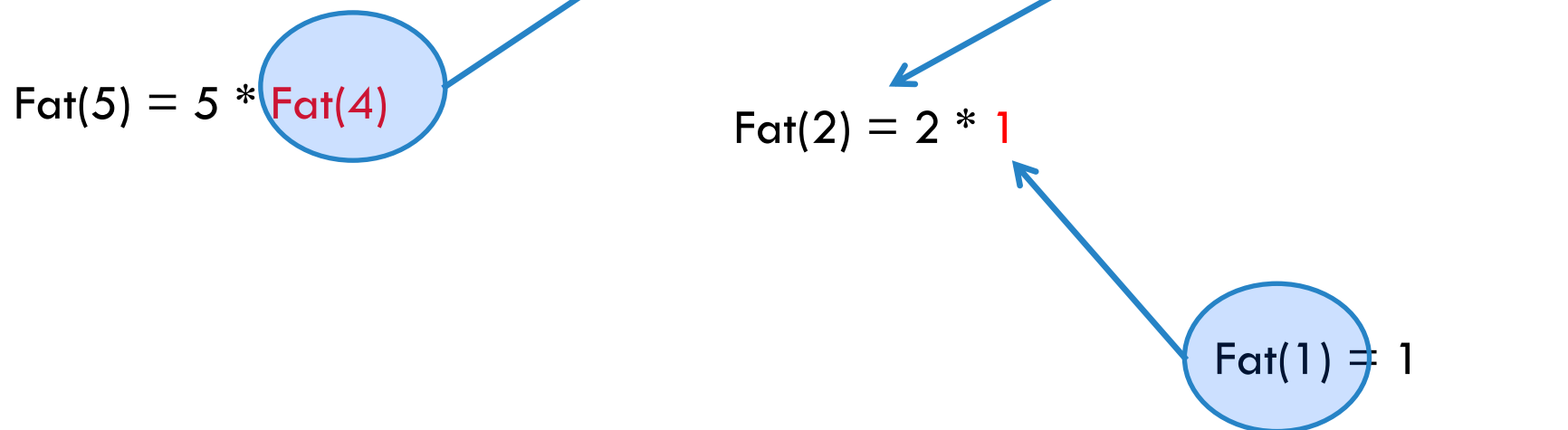


Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

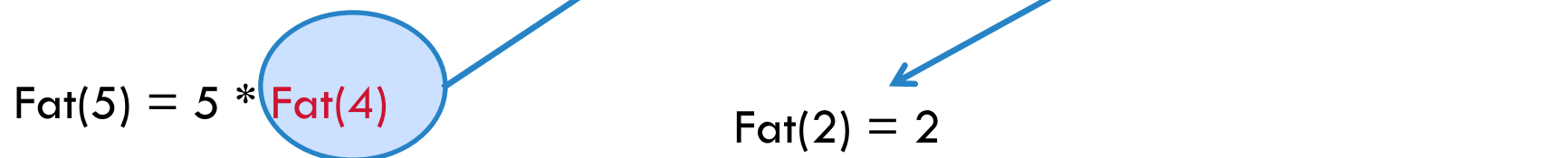


Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?



Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 5 * \text{Fat}(4)$$

$$\text{Fat}(2) = 2$$

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

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

Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 5 * \text{Fat}(4)$$

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

$$\text{Fat}(3) = 6$$

Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?




Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

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

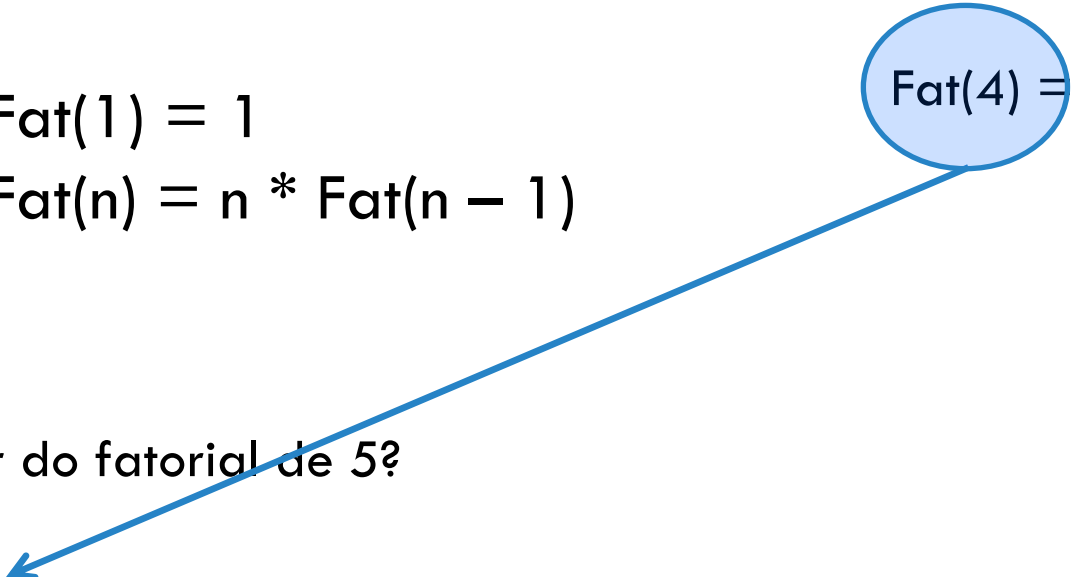
- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 5 * \text{Fat}(4)$$


Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$


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

- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 5 * 24$$

Exemplo: Fatorial Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fat}(1) = 1 \\ \text{Fat}(n) = n * \text{Fat}(n - 1) \end{cases}$$

- Qual é o valor do fatorial de 5?

$$\text{Fat}(5) = 120$$

Exemplo: Fatorial Recursivo

```
int fatorial (int n){  
    int resp;  
    if (n == 1){  
        resp = 1;  
    } else {  
        resp = n * fatorial(n - 1);  
    }  
    return resp;  
}  
void main(){  
    int valor = fatorial(5);  
    printf("%d",valor);  
}
```

Exemplo: Fatorial Recursivo

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

```
void main(){  
    int valor = fatorial(5);  
    printf("%d",valor);  
}
```

Exemplo: Fatorial Recursivo

```
int fatorial (int n){  
    int resp;  
    if (n == 1){  
        resp = 1;  
    } else {  
        resp = n * fatorial(n - 1);  
    }  
    return resp;  
}  
void main(){  
    int valor = fatorial(5);  
    printf("%d",valor);  
}
```

Exemplo: Fatorial Recursivo

```
int fatorial (int n){ // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {             false
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```


Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

$$\text{fatorial}(5) = 5 * \text{fatorial}(4)$$

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

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

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

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

Exemplo: Fatorial Recursivo

```
int fatorial (int n){ // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)



Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)



Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {             false
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)



Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){ // n (3)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (3)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (3)
    int resp;
    if (n == 1){
        resp = 1;
    } else {             false
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (3)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (3)
```

```
    int resp;
```

```
    if (n == 1){
```

```
        resp = 1;
```

```
    } else {
```

```
        resp = n * fatorial(n - 1);
```

```
    }
```

```
    return resp;
```

```
}
```

```
void main(){
```

```
    int valor = fatorial(5);
```

```
    printf("%d",valor);
```

```
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){ // n (2)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (2)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (2)
    int resp;
    if (n == 1){
        resp = 1;
    } else {             false
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (2)
```

```
    int resp;
```

```
    if (n == 1){  
        resp = 1;
```

```
    } else {
```

```
        resp = n * fatorial(n - 1);
```

```
    }
```

```
    return resp;
```

```
}
```

```
void main(){
```

```
    int valor = fatorial(5);
```

```
    printf("%d",valor);
```

```
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (2)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

Exemplo: Fatorial Recursivo

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

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

fatorial (1) = 1

Exemplo: Fatorial Recursivo

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

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

fatorial (1) = 1

Exemplo: Fatorial Recursivo

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

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

fatorial (1) = 1

Exemplo: Fatorial Recursivo

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

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

fatorial (1) = 1

Exemplo: Fatorial Recursivo

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

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * fatorial (1)

fatorial (1) = 1

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (2)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2 * 1

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (2)
```

```
    int resp;
```

```
    if (n == 1){
```

```
        resp = 1;
```

```
    } else {
```

```
        resp = n * fatorial(n - 1);
```

```
    }
```

```
    return resp;
```

```
}
```

```
void main(){
```

```
    int valor = fatorial(5);
```

```
    printf("%d",valor);
```

```
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (2)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * fatorial (2)

fatorial (2) = 2

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (3)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 3 * 2

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (3)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 6

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (3)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * fatorial (3)

fatorial (3) = 6

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 4 * 6


Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 24




Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (4)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 5 * fatorial (4)

fatorial (4) = 24



Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

$$\text{fatorial}(5) = 5 * \boxed{24}$$

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 120

Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 120



Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

fatorial (5) = 120

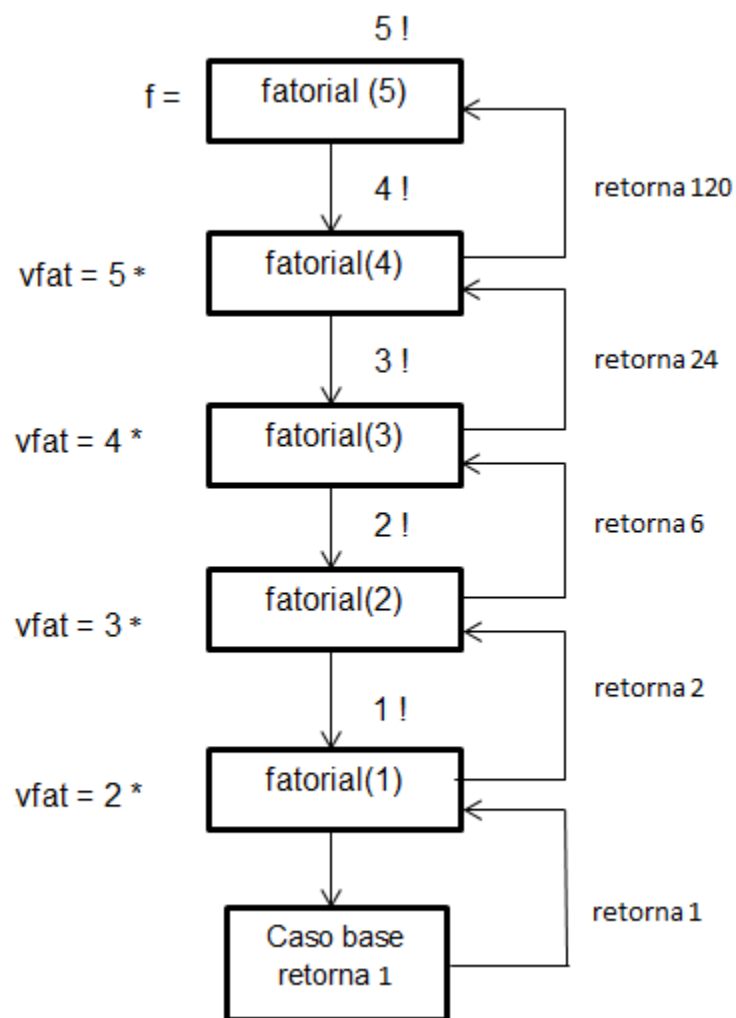


Exemplo: Fatorial Recursivo

```
int fatorial (int n){    // n (5)
    int resp;
    if (n == 1){
        resp = 1;
    } else {
        resp = n * fatorial(n - 1);
    }
    return resp;
}

void main(){
    int valor = fatorial(5);
    printf("%d",valor);
}
```

Exemplo: Fatorial Recursivo



Exemplo 06

Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\left\{ \begin{array}{l} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{array} \right.$$

- Qual é o Fibonacci de 4?

Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\left\{ \begin{array}{l} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{array} \right.$$

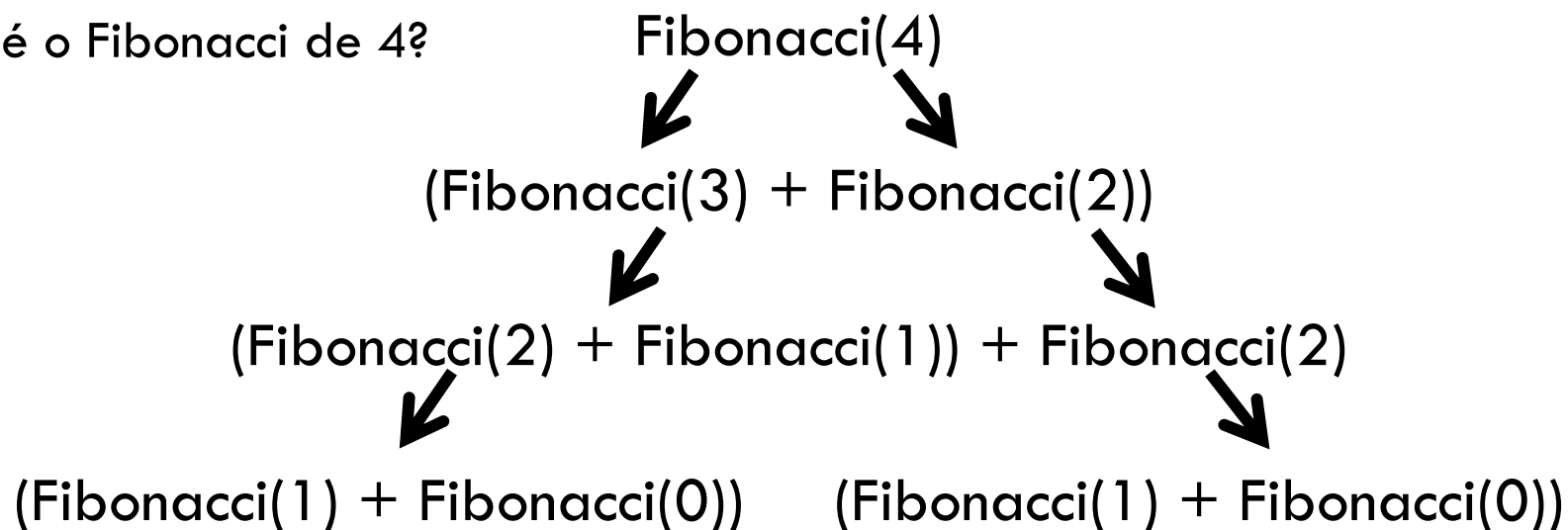
- Qual é o Fibonacci de 4? $\text{Fibonacci}(4)$

Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?

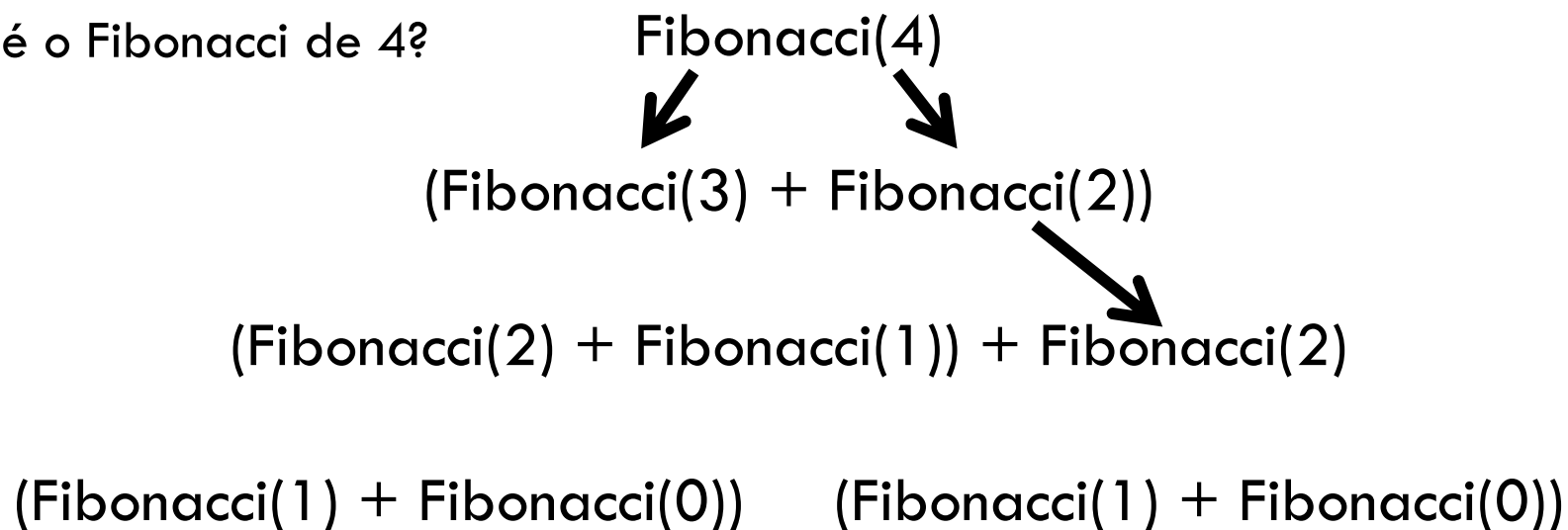


Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?



Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?

$$\begin{array}{c} \text{Fibonacci}(4) \\ \swarrow \quad \searrow \\ (\text{Fibonacci}(3) + \text{Fibonacci}(2)) \\ \\ (\text{Fibonacci}(2) + \text{Fibonacci}(1)) + \text{Fibonacci}(2) \\ \searrow \\ (1 + 1) \end{array}$$

Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\left\{ \begin{array}{l} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{array} \right.$$

- Qual é o Fibonacci de 4?

$$\begin{array}{c} \text{Fibonacci}(4) \\ \swarrow \quad \searrow \\ (\text{Fibonacci}(3) + \text{Fibonacci}(2)) \end{array}$$

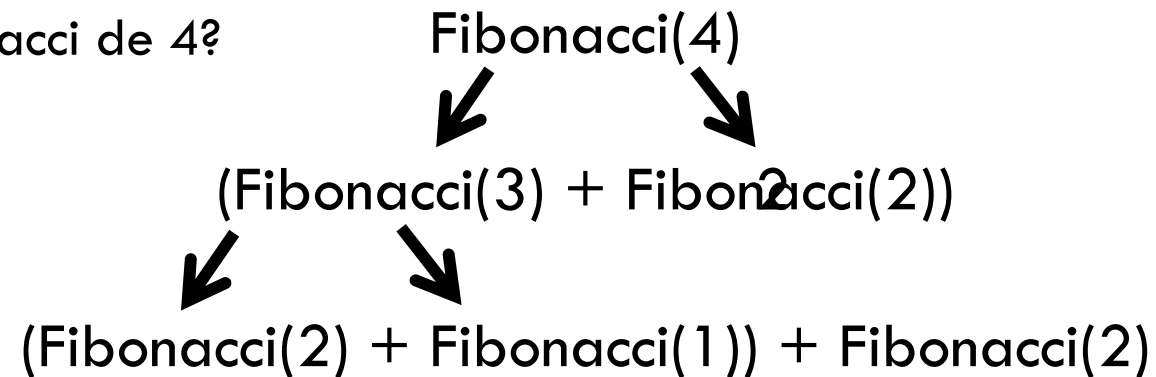
$$(\text{Fibonacci}(2) + \text{Fibonacci}(1)) + \text{Fibonacci}(2)$$

Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?

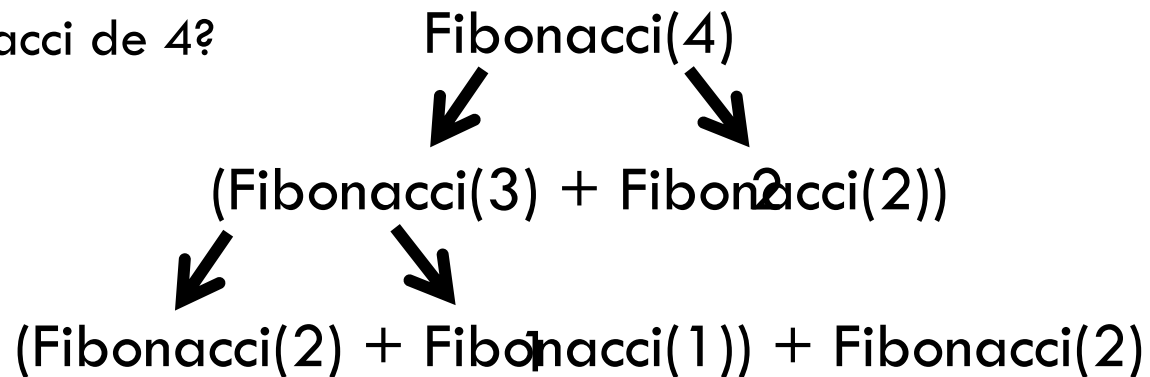


Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?

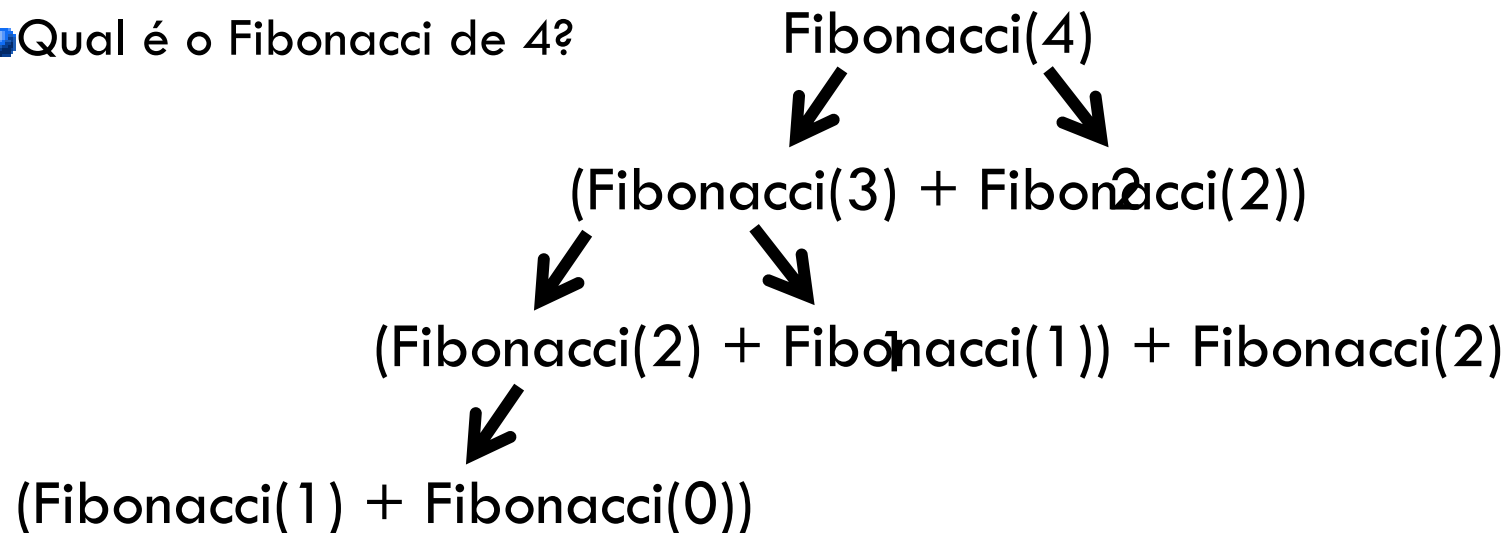


Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?

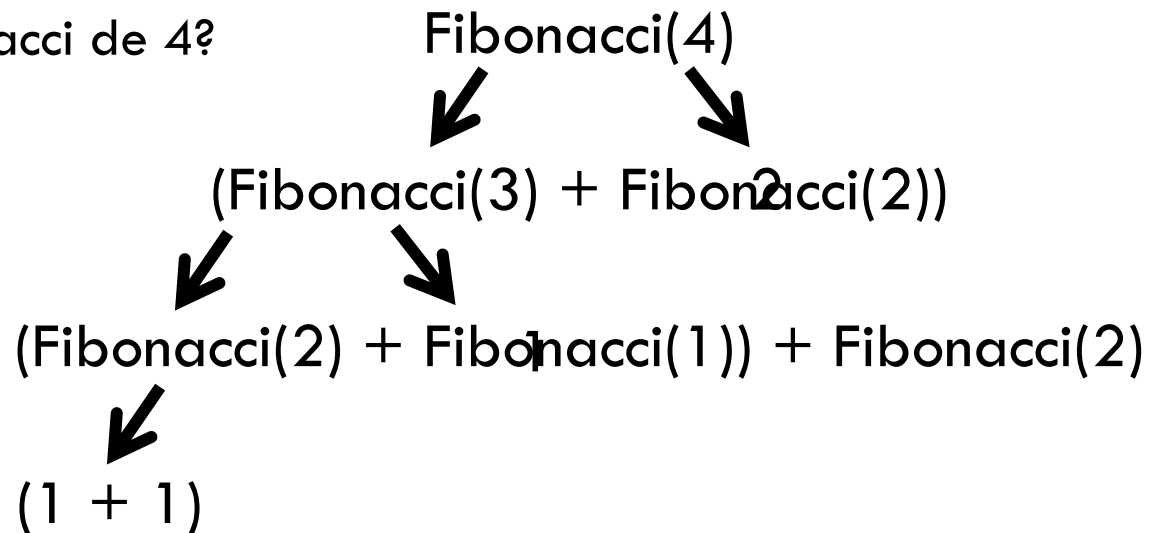


Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?

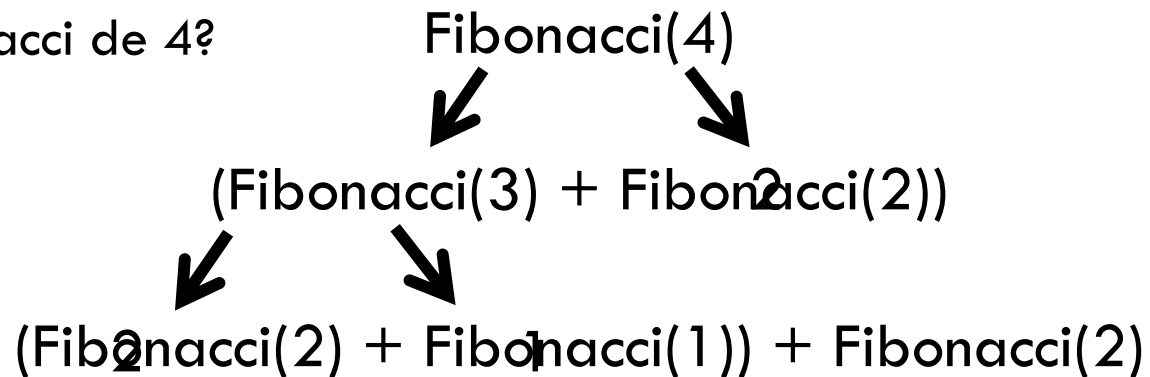


Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\begin{cases} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{cases}$$

- Qual é o Fibonacci de 4?



Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\left\{ \begin{array}{l} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{array} \right.$$

- Qual é o Fibonacci de 4?

$$\begin{array}{c} \text{Fibonacci}(4) \\ \swarrow \quad \searrow \\ (\text{Fibonacci}(3) + \text{Fibonacci}(2)) \end{array}$$

Exemplo: Fibonacci Recursivo

- Definição do fatorial é recursiva:

$$\left\{ \begin{array}{l} \text{Fib}(0) = 1 \\ \text{Fib}(1) = 1 \\ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \end{array} \right.$$

- Qual é o Fibonacci de 4?

5

Exemplo: Fibonacci Recursivo

```
int fibonacci (int n){  
    int resp;  
    if (n == 0 || n == 1){  
        resp = 1;  
    } else {  
        resp = fibonacci(n - 1) + fibonacci(n - 2);  
    }  
    return resp;  
}  
void main(){  
    int valor = fibonacci(4);  
    printf("%d",valor);  
}
```

Exercícios

Exercício

- Faça uma função recursivo que receba dois números inteiros e retorne a multiplicação do primeiro pelo segundo fazendo somas

$$5 \times 13 = \underbrace{5 + 5 + \dots + 5}_{13 \text{ vezes}}$$

Exercício

- Faça uma função recursivo que receba dois números inteiros e retorne a multiplicação do primeiro pelo segundo fazendo somas

```
int multiplicacao (int a, int b){  
    int resp = 0;  
  
    if (b > 0){  
        resp = a + multiplicacao(a, b - 1);  
    }  
  
    return resp;  
}  
void main (...){  
    multiplicacao(4, 3);  
}
```

Exercício

- Faça uma função recursivo que receba dois números inteiros e retorne a multiplicação do primeiro pelo segundo fazendo somas (**outra resposta**)

```
int multiplicacao (int a, int b, int i){  
    int resp = 0;  
  
    if (i < b){  
        resp = a + multiplicacao(a, b, i + 1);  
    }  
  
    return resp;  
}  
  
int multiplicacao (int a, int b){  
    return multiplicacao(a, b, 0);  
}
```

```
int multiplicacao (int a, int b){  
    int resp = 0;  
  
    for (int i = 0; i < b; i = i + 1){  
        resp += a;  
    }  
  
    return resp;  
}
```

Considerações

Considerações

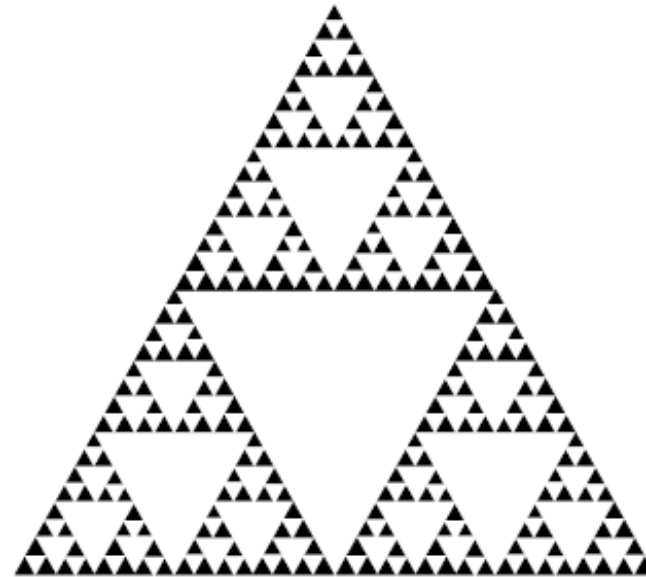
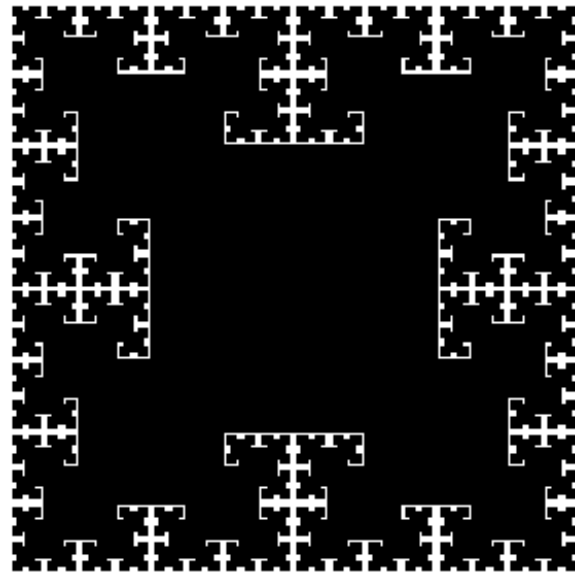
- Todo programa iterativo pode ser feito de forma recursiva e vice-versa.
- Algumas vezes é mais “fácil” fazer um programa de forma recursiva!!!
- O conceito de recursividade é fundamental na computação e na matemática (por exemplo, número naturais, fatorial e outros)
- A recursividade pode ser direta ou indireta (A chama B que chama A)

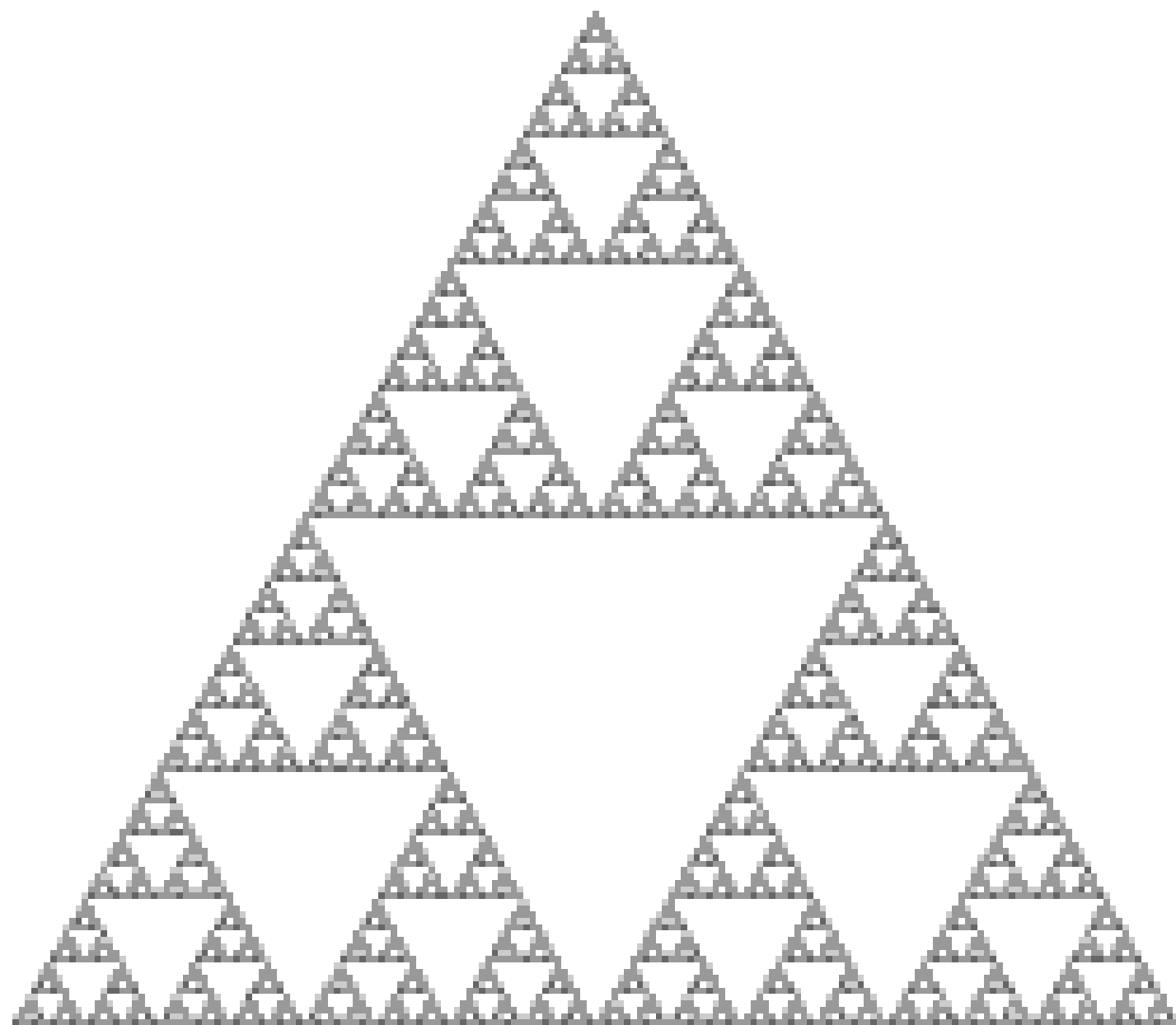
Considerações

- O SO usa uma pilha para armazenar o estado corrente do programa antes de cada chamada não terminada e quando uma chamada termina, o SO recupera o estado armazenado na pilha
- As variáveis locais são recriadas para cada chamada recursiva
- Por que na prática é importante manter um nível “limitado” de chamadas recursivas?

Considerações

- Outro exemplo de recursividade são os fractais, pequenos padrões geométricos que ao serem repetidos diversas vezes de forma recursiva criam desenhos mais sofisticadas





Resumo

A forma de calcular $n!$ é a mesma forma de calcular $(n-1)!$, o que muda são os valores utilizados, ou seja, a mesma lógica se repete!

Recursividade

Pelos dicionários,

“Um objeto é dito recursivo se pode ser definido em termos de si próprio, ou seja, o que é definido aparece na sua própria definição”.

“O que se consegue repetir pela aplicação da mesma regra”.

Vamos lembrar de como funciona o cálculo do fatorial de um número positivo:

$$0! = 1! = 1$$

$$4! = 4 * 3 * 2 * 1 = 24$$

$$5! = 5 * 4 * 3 * 2 * 1 \text{ ou } 5! = 5 * 4! = 120$$

De modo geral,

$$n! = n * (n-1) * (n-2) * \dots * 1 \quad \forall n \geq 0$$

Recursividade

Em algoritmos existem dois tipos de recursão: direta e indireta.

Direta: um módulo tem dentro os seus comandos um acionamento para ele mesmo.

Indireta: um módulo aciona outro que aciona o primeiro em seus comandos, ou seja, A aciona B que aciona A.

Para que a recursão não seja infinita, torna-se necessário incluir uma **condição de parada**.

Deve-se ter cuidado com o uso de módulos recursivos, pois **a cada chamada** de um módulo o seu contexto (parâmetros, variáveis locais, próximo comando) é armazenado na memória e essa área somente será liberada ao término da execução do módulo e retorno ao ponto de chamada.

Assim, para N chamadas, tem-se N vezes a área de memória reservada para sua execução.

Recursividade

Recursividade vale a pena para algoritmos complexos, cuja implementação iterativa é complexa.

Chama-se de **caso base** aquela situação cujo resultado é obtido de modo direto e imediato. E o **passo recursivo** se refere à aplicação recursiva do módulo em uma parte menor do problema.

Outra questão é o contexto ou escopo do módulo. A cada chamada do módulo, cria-se uma **nova área na memória** para armazenar todo o contexto (variáveis, parâmetros, comandos, ponto de retorno) daquela chamada.

Dessa forma, mesmo que os nomes sejam idênticos, eles são **independentes** dos correspondentes da chamada anterior por se encontrarem em áreas distintas.

Recursividade

Todo algoritmo recursivo pode ser escrito de maneira não recursiva ou iterativa.

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

```
double calculaFatorial (int n)
{
    double fatorial = 1;

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

```

1 double fatorialRecursivo (int n)
2 {
3     if (n == 0) || (n == 1)
4     {
5         return (1);
6     }
7     else
8     {
9         return (n*fatorialRecursivo(n-1));
10    }
11 }

```

Se em determinada linha do main, a função fatorialRecursivo é chamada com o argumento 3, temos o seguinte fluxo de execução para a primeira vez (A) que a função é chamada:

Linha A1: fatorialRecursivo (3), n = 3
 Linha A3: false
 Linha A9: return(3 * fatorialRecursivo(2))

Nesse ponto, o fluxo é interrompido para execução de NOVO módulo. Mesmo sendo o mesmo, será uma nova instância daquele módulo (B) e ao término dessa nova instância, retorna para a Linha A9 com o resultado.

Linha B1: fatorialRecursivo (2), n = 2

Linha B3: false

Linha B9: return(2 * fatorialRecursivo(1)) // nova interrupção

Nova chamada (C):

Linha C1: fatorialRecursivo(1), n = 1

Linha C3: true

Linha C4: return 1 // Valor retornado para chamada em B9

Linha B9: return (2*1) => return (2)

// retornado para chamada em A9

Linha A9: return(3 * 2)=> return(6) //retornado para o Main

fibonacci

Sequência de números na qual cada elemento é a soma dos dois anteriores, exceto pelos dois primeiros que são definidos.

Sendo 'n' a **posição** do elemento na sequência de números, tem-se que

$F(n) = n$, para $n == 0$ ou $n == 1$

$F(n) = F(n-1) + F(n-2)$, $\forall n \geq 2$

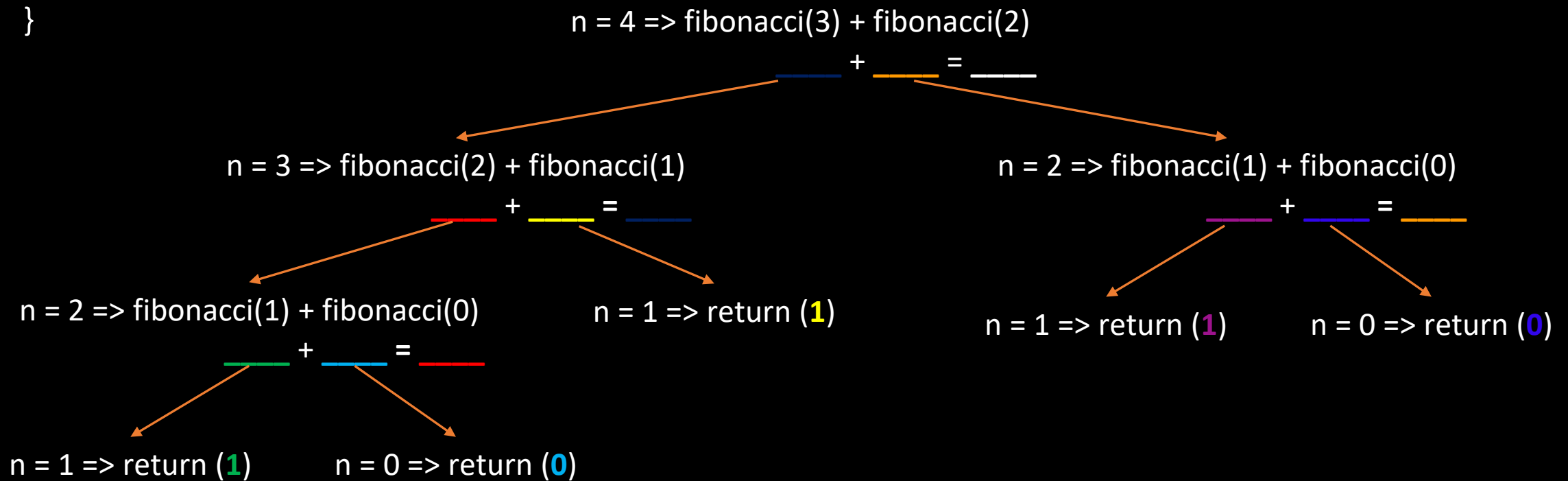
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,...

```
void int fibonacciLoop(int n)
{
    int i, j, termo, proximo;
    i = 0; j = 1; // valores dos dois primeiros
    for (termo = 1; termo <= n; termo++)
    {
        proximo = i + j;
        i = j;
        j = proximo;
    }
    if (n <= 0)
    {
        return i;
    }
    else
    {
        return j;
    }
}
```

```

1  int fibonacciRecursivo (int n)
2  {
3      if (n < 2)
4      {
5          return (n);
6      }
7      else
8      {
9          return (fibonacciRecursivo(n-2) + fibonacciRecursivo(n-1));
10     }
11 }

```



```

1  int fibonacciRecursivo (int n)
2  {
3      if (n < 2)
4      {
5          return (n);
6      }
7      else
8      {
9          return (fibonacciRecursivo(n-2) + fibonacciRecursivo(n-1));
10     }
11 }

```

Se em determinada linha do Main, a função fibonacciRecursivo é chamada com o argumento 4, temos o seguinte fluxo de execução para a primeira vez (A) que a função é chamada:

Linha A1: fibonacciRecursivo(4), n = 4

Linha A3: false

Linha A9: return (fibonacciRecursivo(3) + fibonacciRecursivo(2))

Nesse ponto, o fluxo é interrompido para execução de NOVO módulo para 'n-1' (uma chamada por vez). Mesmo sendo o mesmo, será uma nova instância daquele módulo (B) e ao término dessa nova instância, retorna para a Linha A9 com o resultado e interrompe novamente

para execução de OUTRO módulo para 'n-2' (outra chamada), ou seja, nova instância daquele módulo (?).

Linha B1: fibonacciRecursivo (3), n = 3

Linha B3: false

Linha B9: return(fibonacciRecursivo(2) + fibonacciRecursivo(1)) // interrompe 2x, uma para 2 // e outra para 1

Nova chamada (C) para o termo 2 chamado em B9:

Linha C1: fibonacciRecursivo(2), n = 2

Linha C3: false

Linha C9:

return(fibonacciRecursivo(1)+fibonacciRecursivo(0))
// interrompe 2x, uma para 1 e outra para 0

Nova chamada (D) para o termo 1 chamado em C9:

Linha D1: fibonacciRecursivo(1), n = 1

Linha D3: true

Linha D5: return (1) // retornado para C9

Linha C9: return(1 + fibonacciRecursivo(0))

```

1 public static int fibonacciRecursivo (int n)
2 {
3     if (n < 2)
4     {
5         return (n);
6     }
7     else
8     {
9         return (fibonacciRecursivo(n-2) + fibonacciRecursivo(n-1));
10    }
11 }

```

Nova chamada (E) para o termo 0 chamado em C9:

Linha E1: fibonacciRecursivo(0), n = 0

Linha E3: true

Linha E5: return (0) // retornado para C9

Linha C9: return(1 + 0) => return (1) // retornado para B9

Linha B9: return(1 + fibonacciRecursivo(1))

Nova chamada (F) para o termo 1 chamado em B9:

Linha F1: fibonacciRecursivo(0), n = 0

Linha F3: true

Linha F5: return (1) // retornado para B9

Linha B9: return(1+1) => return (2) // retornado para A9

Linha A9: return(2 + fibonacciRecursivo(2))

Nova chamada (G) para o termo 2 chamado em A9:

Linha G1: fibonacciRecursivo(2), n = 2

Linha G3: false

Linha G9:

return(fibonacciRecursivo(1)+fibonacciRecursivo(0))

// interrompe 2x, uma para 1 e outra para 0

Nova chamada (H) para o termo 1 chamado em G9:

Linha H1: fibonacciRecursivo(1), n = 1

Linha H3: true

Linha H5: return (1) // retornado para G9

Linha G9: return(1 + fibonacciRecursivo(0))

```
1 public static int fibonacciRecursivo (int n)
2 {
3     if (n < 2)
4     {
5         return (n);
6     }
7     else
8     {
9         return (fibonacciRecursivo(n-2) + fibonacciRecursivo(n-1));
10    }
11 }
```

Nova chamada (I) para o termo 0 chamado em G9:

Linha I1: fibonacciRecursivo(0), n = 0

Linha I3: true

Linha I5: return (0) // retornado para G9

Linha G9: return(1 + 0) => return (1) // retornado para A9

Linha A9: return(2+1) => return (3) // retornado para o Main

=> F(4) = 3