

# Estruturas de Dados: **Listas Encadeadas Simples com recursividade**

Helena Graziottin Ribeiro  
[hgrib@ucs.br](mailto:hgrib@ucs.br)

# Função recursiva

- uma função é dita recursiva quando dentro do seu código existe uma chamada para si mesma

```
/* Função imprime recursiva invertida */  
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst_vazia(lst) ) {  
        /* imprime sub-lista */  
        lst_imprime_rec(lst->prox);  
        /* imprime ultimo elemento */  
        printf("info: %d\n",lst->info);  
    }  
}
```

# Função recursiva

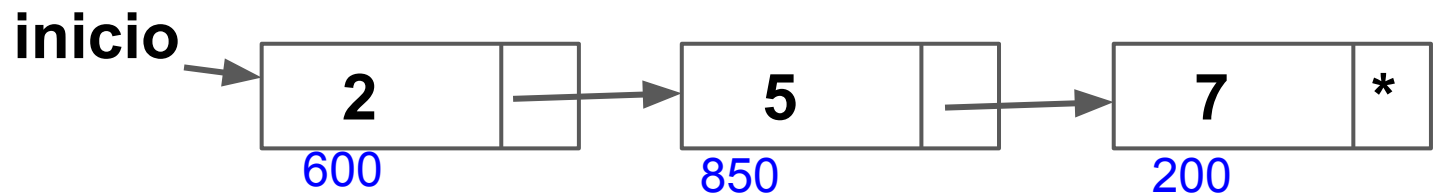
- **computacionalmente elegante**, define funções compactas
- representa uma função com repetição (while, for)
- não apresenta o melhor desempenho:
  - **é mais lenta que uma função com repetição**
  - **ocupa mais espaço na memória**
- algumas soluções complexas exigem implementações recursivas

# Função recursiva para imprimir uma lista

- se a lista for vazia, não imprime nada
- caso contrário,
  - imprime a sub-lista, dada por **lst->prox**, chamando recursivamente a função
  - imprime a informação associada ao primeiro nó, dada por **lst->info**

```
void lst_imprime_rec (Elemento* lst)
{
    if ( !lst_vazia(lst) ) {
        lst_imprime_rec(lst->prox);
        printf("info: %d\n", lst->info);
    }
}
```

# Função recursiva para imprimir uma lista




# Função recursiva para imprimir uma lista

- fundamental que a função tenha um comando para parar as chamadas recursivas, senão a função entra em loop:

```
/* Função imprime recursiva invertida */  
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst_vazia(lst) ) {  
        /* imprime sub-lista */  
        lst_imprime_rec(lst->prox);  
        /* imprime ultimo elemento */  
        printf("info: %d\n", lst->info);  
    }  
}
```

em geral é  
uma  
condição:  
**if**, antes da  
chamada  
recursiva



# Função recursiva para imprimir uma lista

- quando uma função é chamada para a execução, é alocada para sua execução uma área de memória na pilha do sistema:

```
void lst_imprime_rec (Elemento*  
lst)  
{  
    if ( !lst_vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info:  
%d\n", lst->info);  
    }  
}
```

```
main ( ) {  
    ...  
    lst_imprime_rec(inicio);  
    ...
```

inicio= ...

retorno para sistema  
operacional

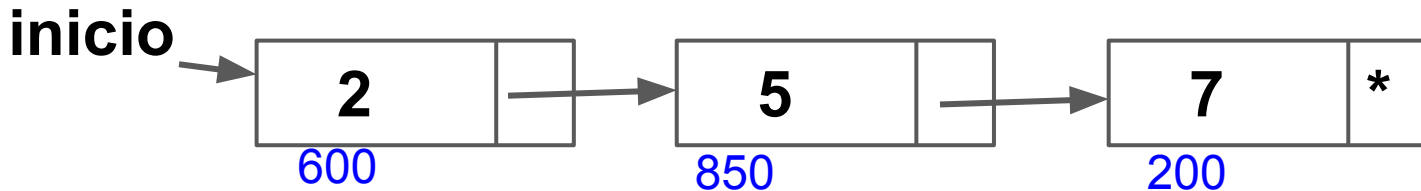
# Função recursiva para imprimir uma lista

- quando uma função é chamada para a execução, é alocada para sua execução uma área de memória na pilha do sistema:

```
int main ( ) {  
...  
lst_imprime_rec(inicio);  
...  
}
```

```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst_vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

lst= 600  
lst->prox= 850  
lst->info= 2  
retorno para main





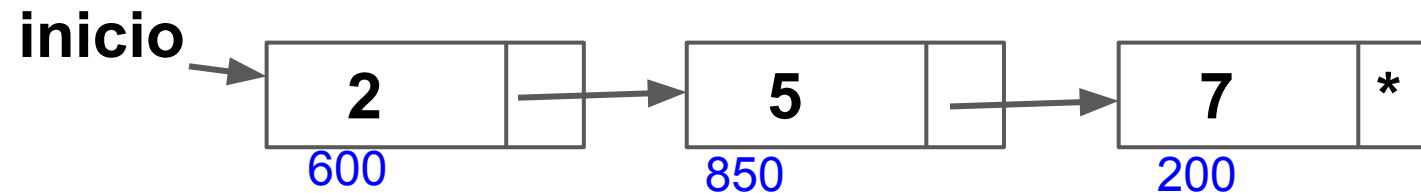
```
int main ( ) {  
...  
lst_imprime_rec(inicio);  
...
```

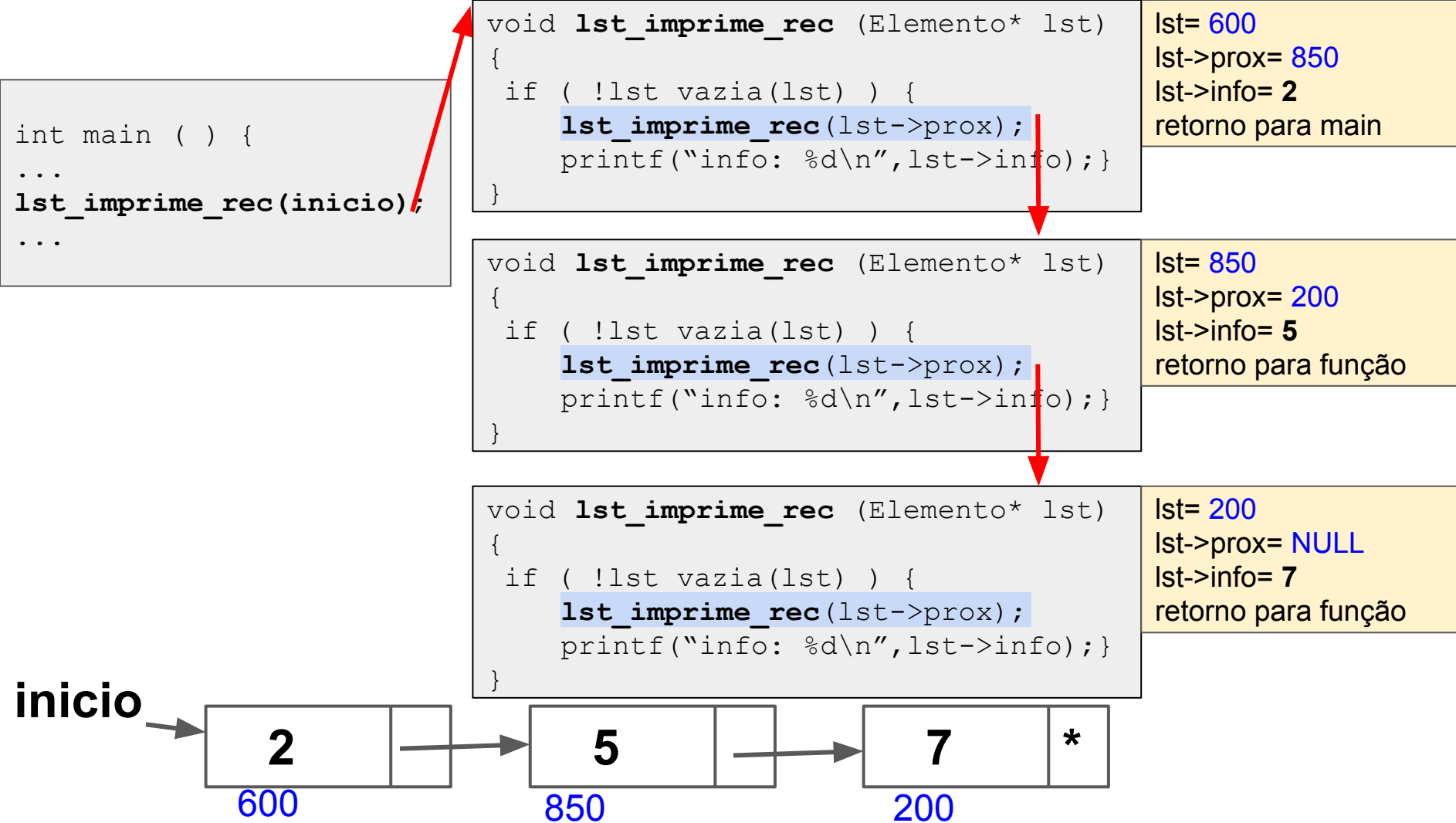
```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst_vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

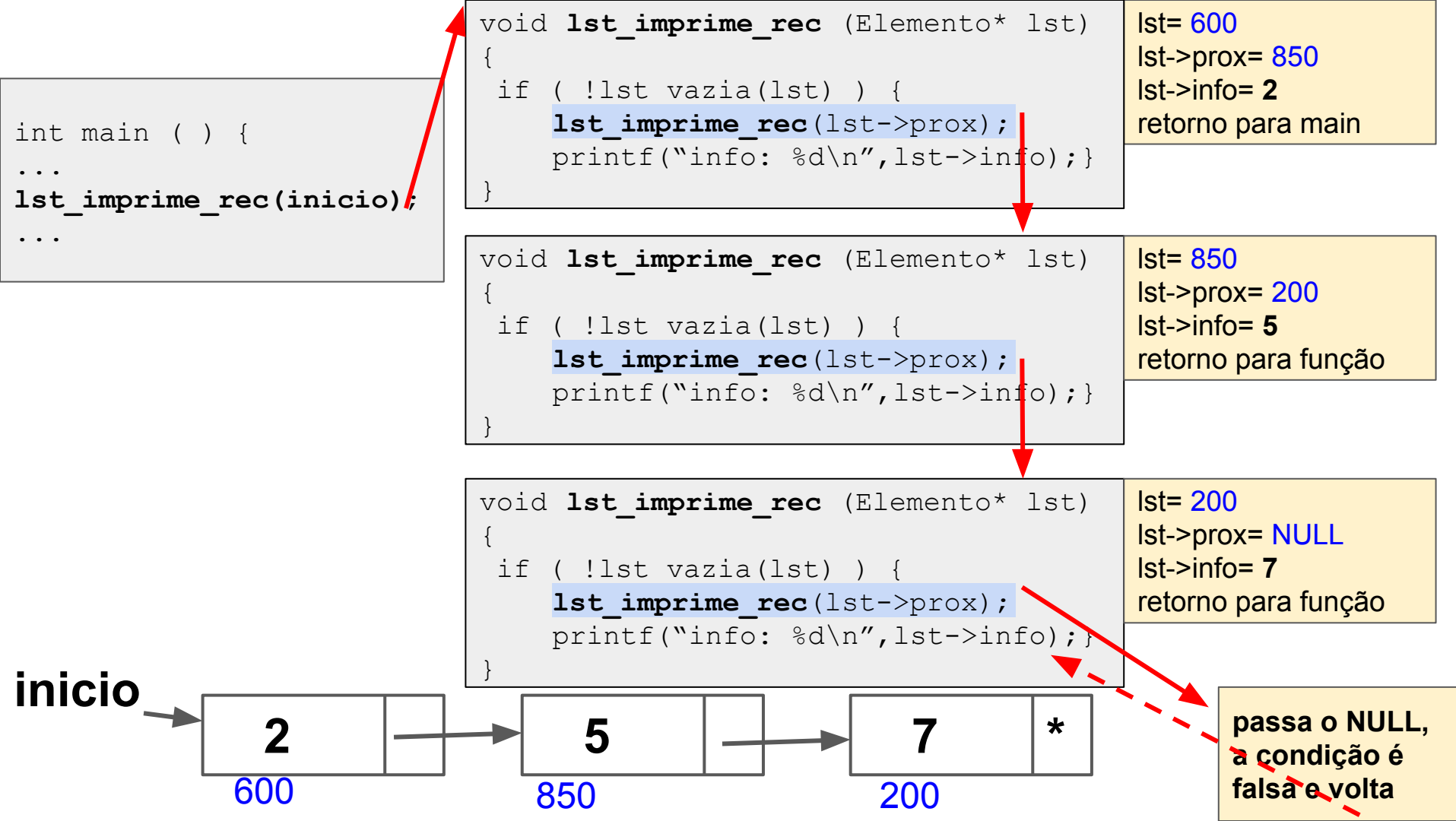
lst= 600  
lst->prox= 850  
lst->info= 2  
retorno para main

```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst_vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

lst= 850  
lst->prox= 200  
lst->info= 5  
retorno para função







```
int main ( ) {  
...  
lst_imprime_rec(inicio);  
...  
}
```

info: 7

```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

lst= 600  
lst->prox= 850  
lst->info= 2  
retorno para main

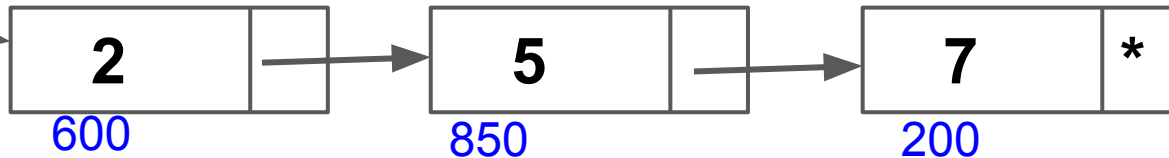
```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

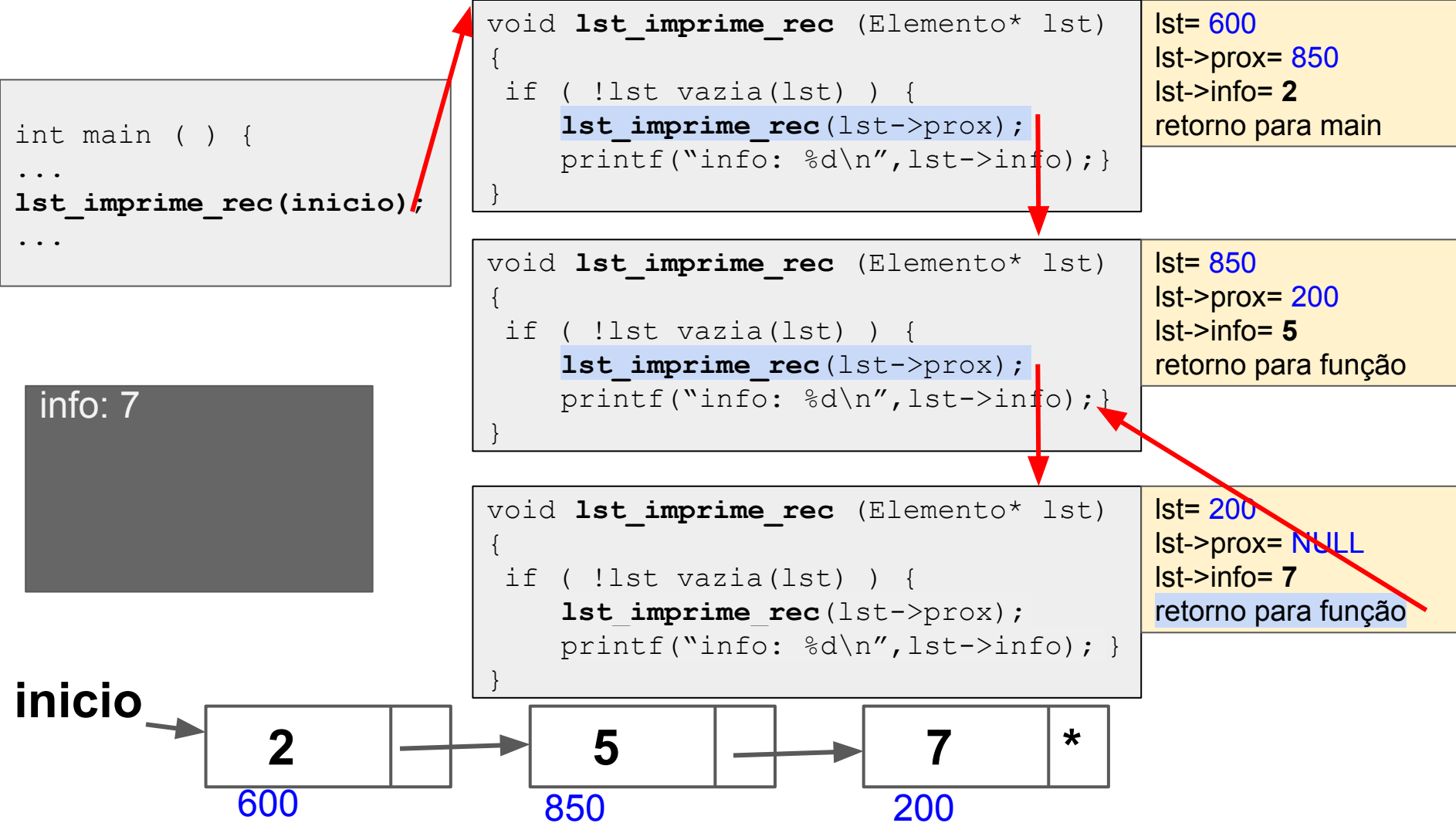
lst= 850  
lst->prox= 200  
lst->info= 5  
retorno para função

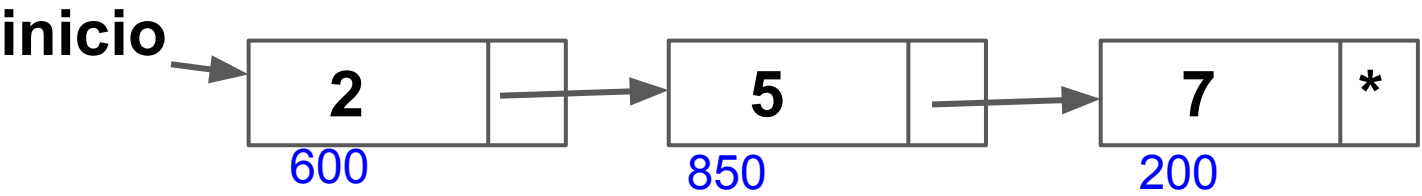
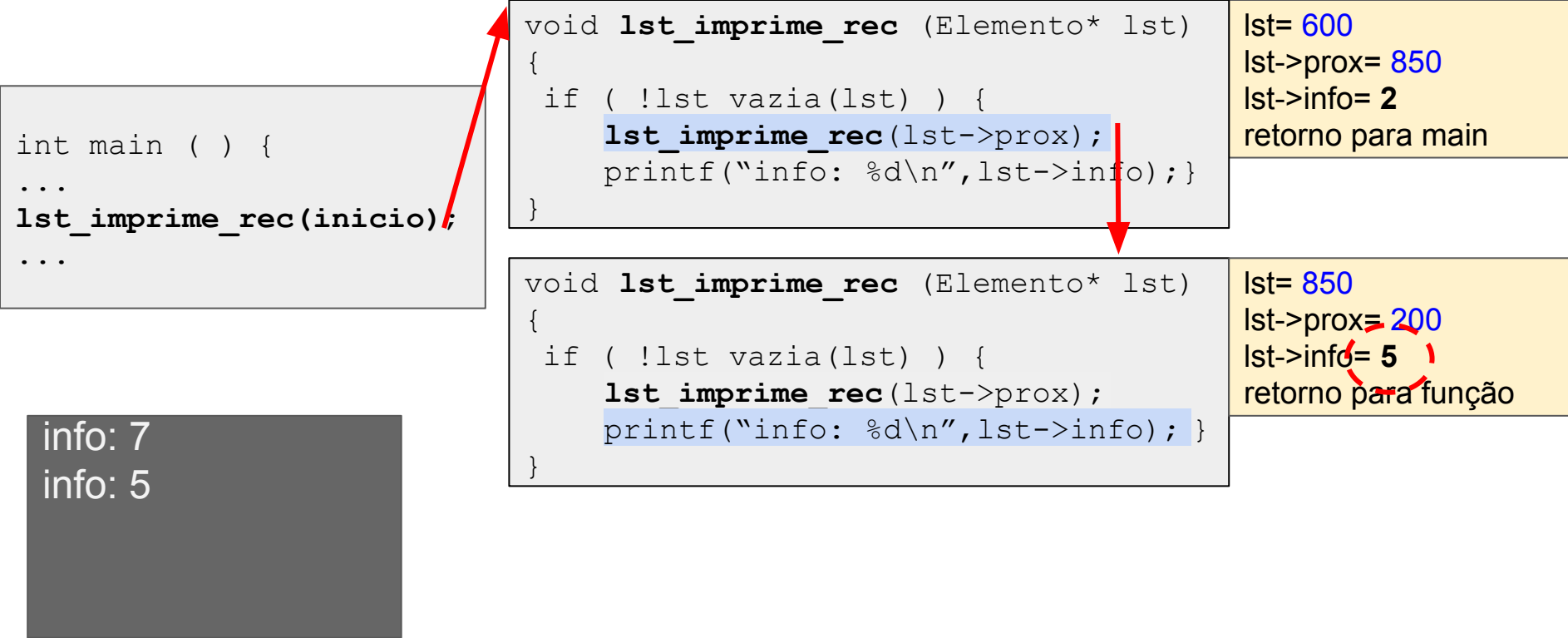
```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

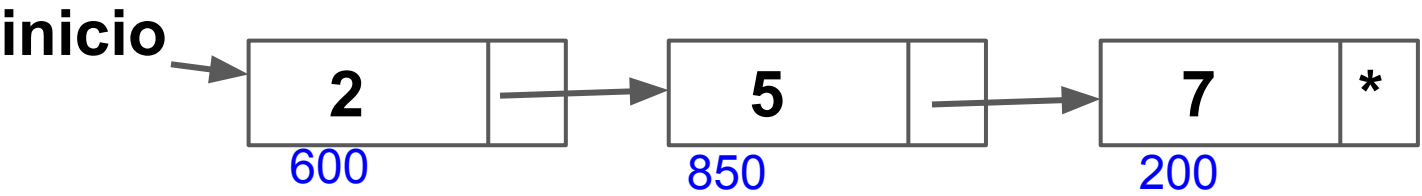
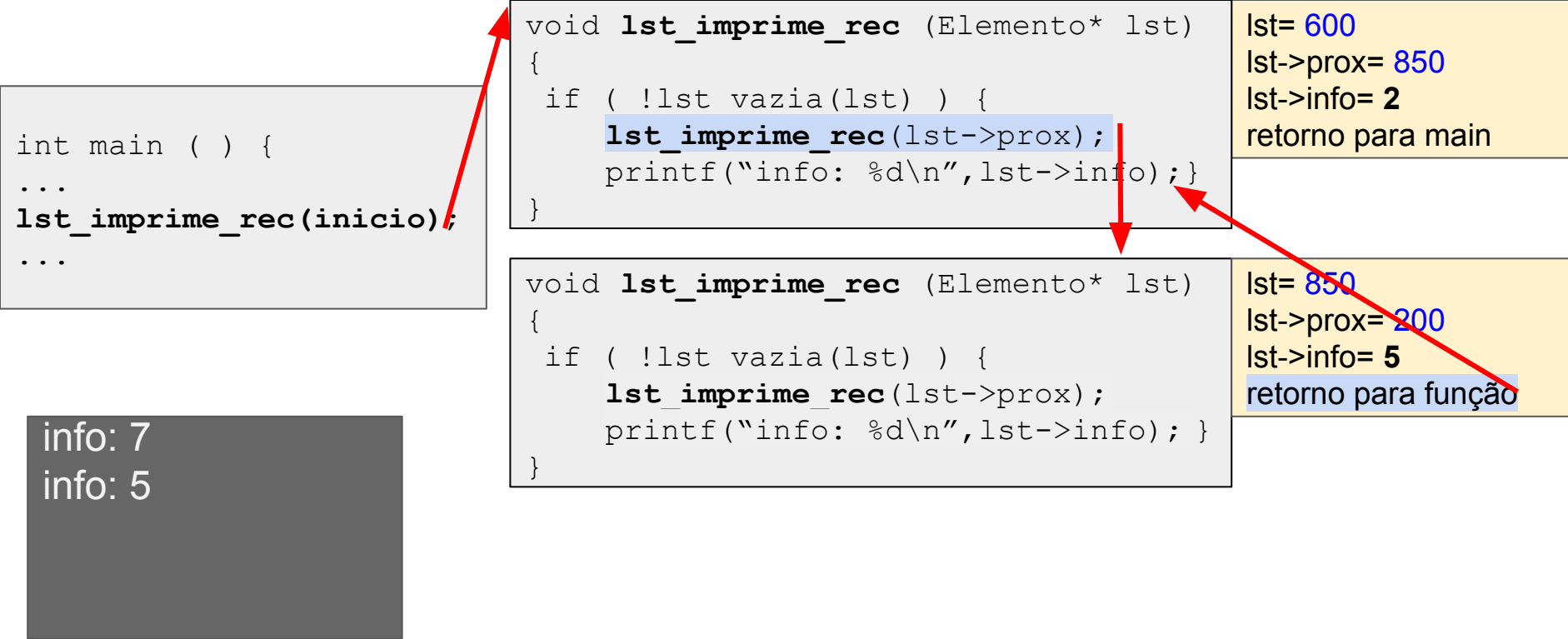
lst= 200  
lst->prox= ~~NULL~~  
lst->info= 7  
retorno para função

inicio







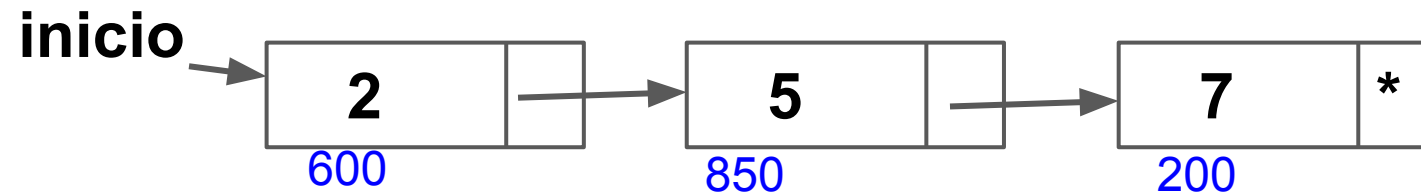


```
int main ( ) {  
...  
lst_imprime_rec(inicio);  
...
```

```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

lst= 600  
lst->prox= 850  
lst->info= 2  
retorno para main

info: 7  
info: 5  
info: 2





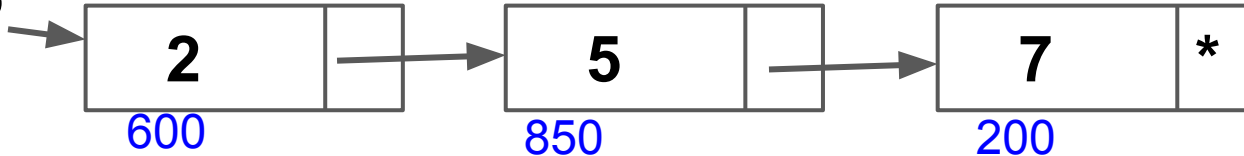
```
int main ( ) {  
...  
lst_imprime_rec(inicio);  
...
```

```
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst vazia(lst) ) {  
        lst_imprime_rec(lst->prox);  
        printf("info: %d\n",lst->info);  
    }  
}
```

lst= 600  
lst->prox= 850  
lst->info= 2  
retorno para main

info: 7  
info: 5  
info: 2

**inicio**



# Função recursiva para imprimir uma lista



info: 2  
info: 5  
info: 7

```
/* Função imprime recursiva */  
void lst_imprime_rec (Elemento* lst)  
{  
    if ( !lst_vazia(lst) ) {  
        /* imprime primeiro elemento */  
        printf("info: %d\n",lst->info);  
        /* imprime sub-lista */  
        lst_imprime_rec(lst->prox);  
    }  
}
```

