Estrutura de Dados I

Estrutura de dados: lista duplamente encadeada

Prof. Rodrigo Minetto

Universidade Tecnológica Federal do Paraná

Sumário

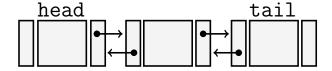
- Introdução
- 2 Função: create
- 3 Função: insert-front
- 4 Função: inserir-front (funcionamento)
- 5 Função: remove-front

A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.



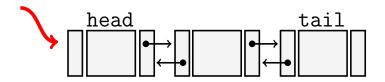
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Pilha:



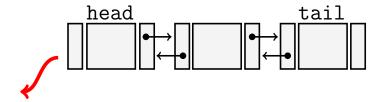
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Pilha: push (cabeça da lista)



A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Pilha: pop (cabeça da lista)



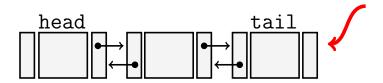
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Fila:



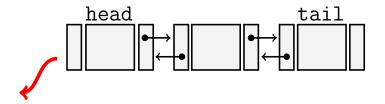
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Fila: enqueue (cauda da lista)



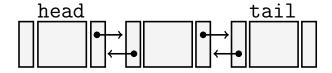
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Fila: dequeue (cabeça da lista)



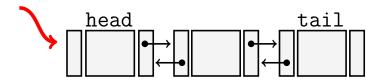
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Deque:



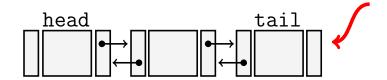
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Deque: insert (cabeça da lista)



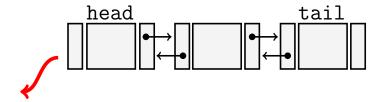
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Deque: insert (cauda da lista)



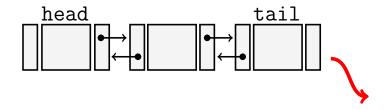
A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

Deque: remove (cabeça da lista)

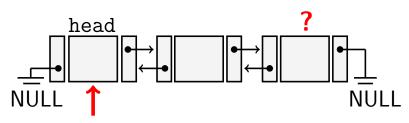


A estrutura de dados **lista** com **encadeamento duplo** é excelente para construir diversas outras estrutura de dados.

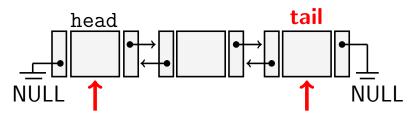
Deque: remove (cauda da lista)



No entanto, a lista com duplo encadeamento vista aula passada **não é eficiente** por exemplo para operações de cauda (complexidade $\mathcal{O}(n)$). Observe que o único ponteiro direto é para a cabeça da lista (**sem acesso direto a cauda**):

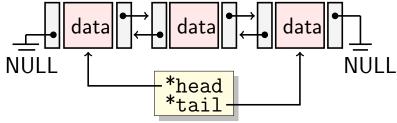


Solução: acrescentar e manipular um ponteiro com acesso direto a cauda da lista. A complexidade para operações de cauda (inserção, busca, remoção de elementos, ...) com esta otimização é $\mathcal{O}(1)$.



Estrutura

```
typedef struct node {
  int data;
    struct node* next;
    struct node* prev;
} List;
} Node;
```



Tipo abstrato de dados

Lista duplamente encadeada (interface) create: inicializa uma estrutura de dados lista insert-front: adiciona um nó no início insert-back: adiciona um nó no fim remove-front: retira um nó do início remove-back: retira um nó do fim **print-front-back**: imprime nós do início p/ fim **print-back-front**: imprime nós do fim p/início

Sumário

- Introdução
- 2 Função: create
- 3 Função: insert-front
- 4 Função: inserir-front (funcionamento)
- 5 Função: remove-front

```
1: List* create (void)
```

- 2: List* $\ell = (List*)malloc(sizeof(List));$
- 3: $\ell \rightarrow \mathsf{head} = \mathsf{NULL};$
- 4: $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$
- 5: return ℓ ;

```
*\ell \frac{\text{*head}}{\text{*tail}}
```

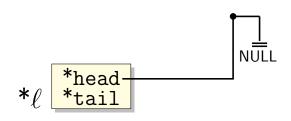
```
1: List* create (void)
```

```
2: List* \ell = (List*)malloc(sizeof(List));

3: \ell \rightarrowhead = NULL;

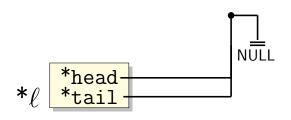
4: \ell \rightarrowtail = NULL;

5: return \ell;
```



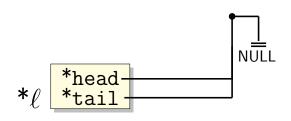
```
1: List* create (void)
```

- 2: List* $\ell = (List*)malloc(sizeof(List));$
- 3: $\ell \rightarrow \text{head} = \text{NULL};$
- 4: $\ell \rightarrow tail = NULL;$
- 5: return ℓ ;



```
1: List* create (void)
```

- 2: List* $\ell = (List*)malloc(sizeof(List));$
- 3: $\ell \rightarrow \text{head} = \text{NULL};$
- 4: $\ell \rightarrow tail = NULL;$
- 5: return ℓ ;

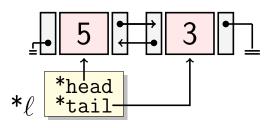


```
1: List* create (void)
```

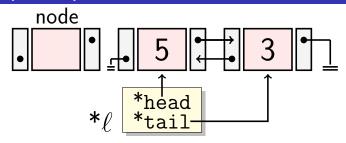
- 2: List* $\ell = (List*)$ malloc(sizeof(List));
- 3: $\ell \rightarrow \text{head} = \text{NULL};$
- 4: $\ell \rightarrow tail = NULL;$
- 5: return ℓ ;

Sumário

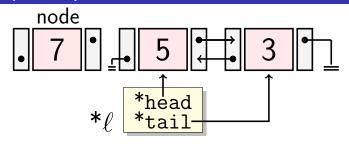
- Introdução
- Punção: create
- 3 Função: insert-front
- 4 Função: inserir-front (funcionamento)
- 5 Função: remove-front



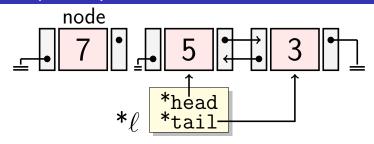
- 1: **void insert-front** (List *ℓ, int elem)
- 2: Node* node = (Node*)malloc(sizeof(Node));
- 3: $node \rightarrow data = elem;$
- 4: $node \rightarrow prev = NULL$;
- 5: ...



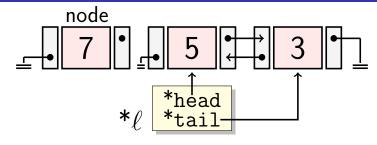
- 1: **void insert-front** (List *ℓ, int elem)
- 2: Node* node = (Node*)malloc(sizeof(Node));
- 3: $node \rightarrow data = elem;$
- 4: $node \rightarrow prev = NULL$;
- 5: ...



- 1: **void insert-front** (List *ℓ, int elem)
- 2: Node* node = (Node*)malloc(sizeof(Node));
- 3: $node \rightarrow data = elem;$
- 4: $node \rightarrow prev = NULL$;
- 5: . . .



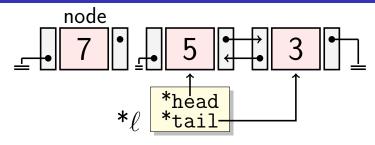
- 1: **void insert-front** (List *ℓ, int elem)
- 2: Node* node = (Node*)malloc(sizeof(Node));
- 3: $node \rightarrow data = elem;$
- 4: $node \rightarrow prev = NULL;$
- 5: . . .



- 1: **void insert-front** (List *\ell, int elem)
- 2: **if** $\ell \rightarrow \text{head} \neq \text{NULL}$;
- 3: $node \rightarrow next = \ell \rightarrow head;$
- 4: $\ell \rightarrow \mathsf{head} \rightarrow \mathsf{prev} = \mathsf{node};$
- 5: **else**

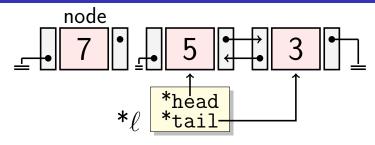
8:

- 6: $node \rightarrow next = NULL$;
- 7: $\ell \rightarrow \mathsf{tail} = \mathsf{node};$
 - $\ell \rightarrow \mathsf{head} = \mathsf{node};$



```
1: void insert-front (List *\ell, int elem)
```

- 2: **if** $\ell \rightarrow \text{head} \neq \text{NULL}$;
- 3: $node \rightarrow next = \ell \rightarrow head;$
- 4: $\ell \rightarrow \mathsf{head} \rightarrow \mathsf{prev} = \mathsf{node};$
- 5: **else**
- 6: $node \rightarrow next = NULL$;
- 7: $\ell \rightarrow \mathsf{tail} = \mathsf{node};$
- 8: $\ell \rightarrow \mathsf{head} = \mathsf{node};$



```
1: void insert-front (List *\ell, int elem)
```

```
if \ell \rightarrow \text{head} \neq \text{NULL};
```

3:
$$node \rightarrow next = \ell \rightarrow head;$$

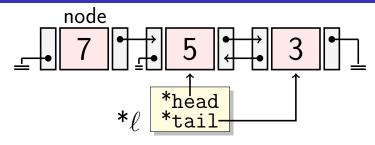
4:
$$\ell \rightarrow \mathsf{head} \rightarrow \mathsf{prev} = \mathsf{node};$$

8:

6:
$$node \rightarrow next = NULL$$
;

7:
$$\ell \rightarrow \mathsf{tail} = \mathsf{node};$$

$$\ell \rightarrow \mathsf{head} = \mathsf{node};$$

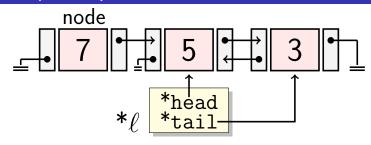


```
1: void insert-front (List *ℓ, int elem)
```

- if $\ell \rightarrow \text{head} \neq \text{NULL}$;
- 3: $node \rightarrow next = \ell \rightarrow head;$
- 4: $\ell \rightarrow \mathsf{head} \rightarrow \mathsf{prev} = \mathsf{node};$
- 5: **else**

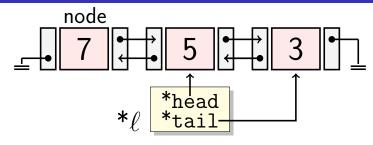
8:

- 6: $node \rightarrow next = NULL$;
- 7: $\ell \rightarrow \mathsf{tail} = \mathsf{node};$
 - $\ell \rightarrow \mathsf{head} = \mathsf{node};$



```
1: void insert-front (List *\ell, int elem)
```

- if $\ell \rightarrow \text{head} \neq \text{NULL}$;
- 3: $node \rightarrow next = \ell \rightarrow head;$
- 4: $\ell \rightarrow \text{head} \rightarrow \text{prev} = \text{node};$
- 5: **else**
- 6: $node \rightarrow next = NULL$;
- 7: $\ell \rightarrow \mathsf{tail} = \mathsf{node};$
- 8: $\ell \rightarrow \mathsf{head} = \mathsf{node};$



```
1: void insert-front (List *\ell, int elem)
```

```
if \ell \rightarrow \text{head} \neq \text{NULL};
```

3:
$$node \rightarrow next = \ell \rightarrow head;$$

4:
$$\ell \rightarrow \text{head} \rightarrow \text{prev} = \text{node};$$

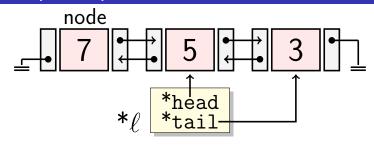
5:

8:

6:
$$node \rightarrow next = NULL$$
;

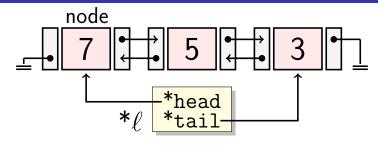
7:
$$\ell \rightarrow \mathsf{tail} = \mathsf{node};$$

$$\ell \rightarrow \mathsf{head} = \mathsf{node};$$



- 1: **void insert-front** (List *\ell, int elem)
- if $\ell \rightarrow \text{head} \neq \text{NULL}$;
- 3: $node \rightarrow next = \ell \rightarrow head;$
- 4: $\ell \rightarrow \mathsf{head} \rightarrow \mathsf{prev} = \mathsf{node};$
- 5: **else**
- 6: $node \rightarrow next = NULL;$
- 7: $\ell \rightarrow \mathsf{tail} = \mathsf{node};$
- 8: $\ell \rightarrow \mathsf{head} = \mathsf{node};$

Inserir (início)



```
1: void insert-front (List *\ell, int elem)
```

- 2: **if** $\ell \rightarrow \text{head} \neq \text{NULL}$;
- 3: $node \rightarrow next = \ell \rightarrow head;$
- 4: $\ell \rightarrow \mathsf{head} \rightarrow \mathsf{prev} = \mathsf{node};$
- 5: **else**
- 6: $node \rightarrow next = NULL$;
- 7: $\ell \rightarrow \mathsf{tail} = \mathsf{node};$
- 8: $\ell \rightarrow \mathsf{head} = \mathsf{node};$

Inserir: 7

Sumário

- Introdução
- 2 Função: create
- 3 Função: insert-front
- 4 Função: inserir-front (funcionamento)
- 5 Função: remove-front

```
1: int main (void)
```

- 2: List * ℓ = create ();
- 3: insert-front $(\ell, 3)$;
- 4: insert-front $(\ell, 5)$;
- 5: insert-front $(\ell, 7)$;
- 6: **return** 0

```
*\ell *head *tail
```

```
1: int main (void)
```

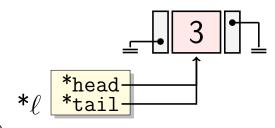
```
2: List *\ell = create ();

3: insert-front (\ell, 3);

4: insert-front (\ell, 5);

5: insert-front (\ell, 7);
```

6: return 0



```
1: int main (void)
```

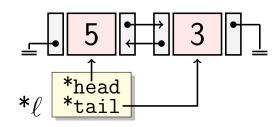
```
2: List *\ell = create ();

3: insert-front (\ell, 3);

4: insert-front (\ell, 5);

5: insert-front (\ell, 7);
```

6: return 0



```
1: int main (void)
```

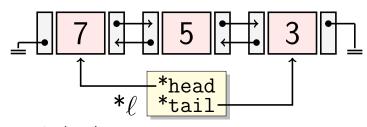
```
2: List *\ell = create ();

3: insert-front (\ell, 3);

4: insert-front (\ell, 5);

5: insert-front (\ell, 7);

6: return 0
```



```
1: int main (void)
```

```
2: List *\ell = create ();

3: insert-front (\ell, 3);

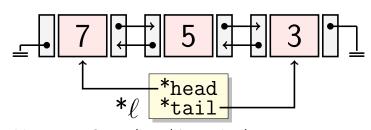
4: insert-front (\ell, 5);

5: insert-front (\ell, 7);

6: return 0
```

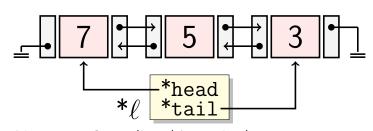
Sumário

- Introdução
- 2 Função: create
- 3 Função: insert-front
- 4 Função: inserir-front (funcionamento)
- 5 Função: remove-front



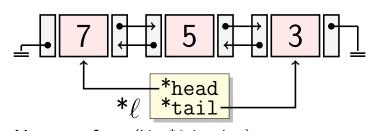
```
1: void remove-front (List *ℓ, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- o: II→prev = NOLL
 7: **else**
- o de la la NII
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



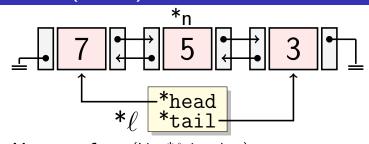
```
1: void remove-front (List *ℓ, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$
- 9: $\ell \rightarrow \text{head} = \text{n}$:



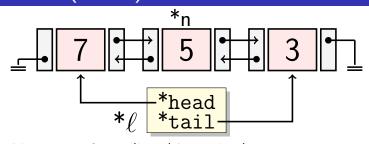
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: else
- i. Cisc
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



```
1: void remove-front (List *\ell, int elem)
```

- if $(\ell \rightarrow head \neq NULL)$ 2:
- Node *n = $\ell \rightarrow head \rightarrow next$; 3:
- **free** ($\ell \rightarrow \text{head}$); 4:
- if $(n \neq NULL)$ 5:
- $n \rightarrow prev = NULL$; 6:
- else 7:
- $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$ 8:
- $\ell \rightarrow \mathsf{head} = \mathsf{n}$: 9:

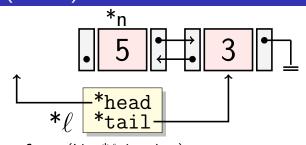


```
1: void remove-front (List *\ell, int elem)
```

- if $(\ell \rightarrow head \neq NULL)$ 2:
- Node *n = $\ell \rightarrow head \rightarrow next$; 3:
- **free** ($\ell \rightarrow \text{head}$); 4:
- if $(n \neq NULL)$ 5:
- $n \rightarrow prev = NULL$; 6:
- else 7:

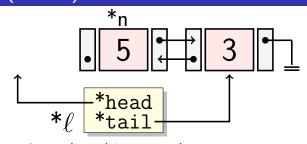
9:

- $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$ 8:
 - $\ell \rightarrow \mathsf{head} = \mathsf{n}$:



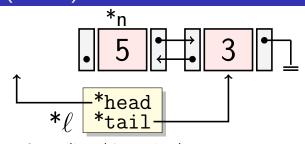
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7· else
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



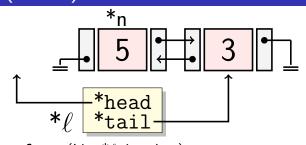
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** $(\ell \rightarrow \text{head})$;
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



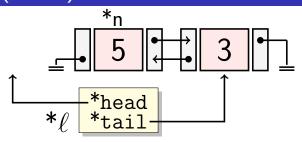
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n}$:



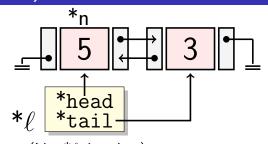
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \text{head} = \text{n}$:



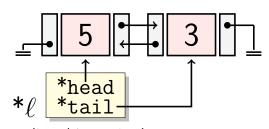
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7· else
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



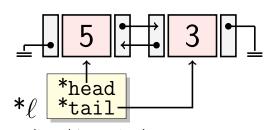
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 7. CISC
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



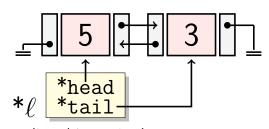
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow head \neq NULL)$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



Remover: 5

```
1: void remove-front (List *\ell, int elem)
```

```
if (\ell \rightarrow head \neq NULL)
2:
```

Node *n =
$$\ell \rightarrow head \rightarrow next$$
;

4: **free** (
$$\ell \rightarrow \text{head}$$
);

3:

7:

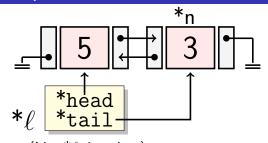
9:

5: **if**
$$(n \neq NULL)$$

6:
$$n \rightarrow prev = NULL$$
;

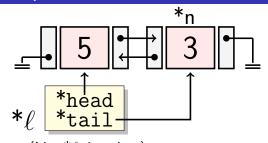
8:
$$\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$$

$$\ell \rightarrow \mathsf{head} = \mathsf{n};$$



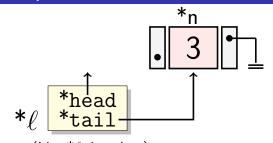
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \text{head} = \text{n}$:



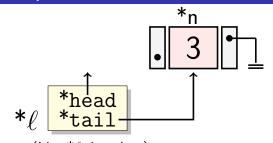
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7· else
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \text{head} = n$;



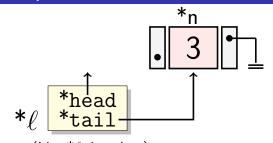
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- Node *n = $\ell \rightarrow \text{head} \rightarrow \text{next}$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n}$:

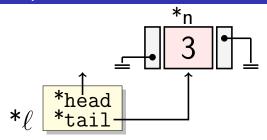


```
1: void remove-front (List *\ell, int elem)
```

- if $(\ell \rightarrow head \neq NULL)$ 2:
- Node *n = $\ell \rightarrow head \rightarrow next$; 3:
- **free** ($\ell \rightarrow \text{head}$); 4: if $(n \neq NULL)$

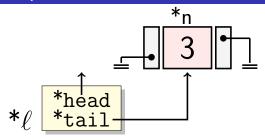
5:

- $n \rightarrow prev = NULL$; 6:
- else 7:
- $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$ 8:
- $\ell \rightarrow \mathsf{head} = \mathsf{n}$: 9:



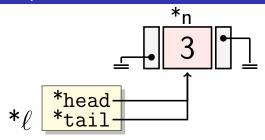
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$:
- $1000e^{-11} = \epsilon 711eau 711ext$
- 4: **free** $(\ell \rightarrow \text{head})$;
- 5: **if** $(n \neq NULL)$ 6: $n \rightarrow prev = NULL$;
- 6: $n\rightarrow prev = NULL;$ 7: **else**
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n}$:



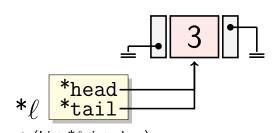
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- free (// vleed):
- 4: **free** $(\ell \rightarrow \text{head})$;
- 5: **if** $(n \neq NULL)$ 6: $n \rightarrow prev = NULL$;
 - $0: \qquad \mathsf{n} \to \mathsf{prev} = \mathsf{NOLL};$
- 7: **else** 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \text{head} = \text{n}$:



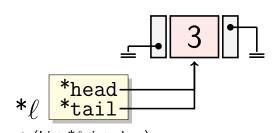
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- o: $n \rightarrow prev = NOLL$;
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \text{head} = \text{n}$:



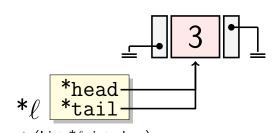
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- o: n→prev = NOLL, 7· **else**
- 7: **else**
- 8: $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



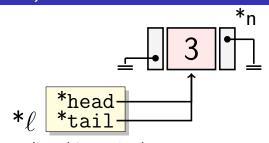
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



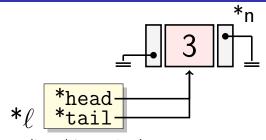
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7· else
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n}$:



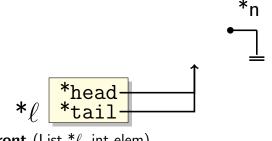
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



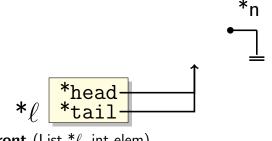
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 4: **free** $(\ell \rightarrow \text{nead})$; 5: **if** $(n \neq \text{NULL})$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



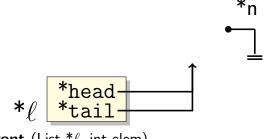
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$

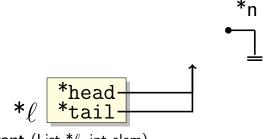


```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow \text{head} \rightarrow \text{next}$;
- 4: **free** $(\ell \rightarrow \text{head})$;
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$

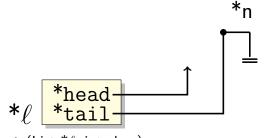


- 1: **void remove-front** (List *\ell, int elem)
- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** $(\ell \rightarrow \text{head})$;
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL;$
- 7: else
- 8: $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



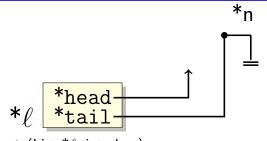
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



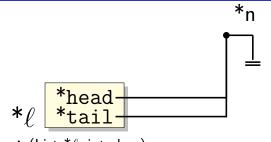
```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL$;
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow head \rightarrow next$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7: **else**
- 8: $\ell \rightarrow tail = NULL;$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$



```
1: void remove-front (List *\ell, int elem)
```

- 2: **if** $(\ell \rightarrow \text{head} \neq \text{NULL})$
- 3: Node *n = $\ell \rightarrow \text{head} \rightarrow \text{next}$;
- 4: **free** ($\ell \rightarrow \text{head}$);
- 5: **if** $(n \neq NULL)$
- 6: $n \rightarrow prev = NULL$;
- 7· else
- 8: $\ell \rightarrow \mathsf{tail} = \mathsf{NULL};$
- 9: $\ell \rightarrow \mathsf{head} = \mathsf{n};$