



UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - CSHNB
CURSO DE SISTEMAS DE INFORMAÇÃO
PICOS - PI

FILAS (ESTÁTICAS E DINÂMICAS)

Prof. Ma. Luana Batista da Cruz
luana.b.cruz@nca.ufma.br

Roteiro

2

- Introdução
- Filas de dados
- Fila estática
- Fila dinâmica

3

Introdução

Definição de fila sequencial

Introdução

4

❑ O que são filas sequenciais?

- São estruturas de dados que armazenam os elementos em um formato sequencial, enfileirando um item atrás do outro

Introdução

5

- **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila do banco



Introdução

6

- **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila do banco



Introdução

7

- **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila do banco



Introdução

8

- ❑ **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila do banco



Introdução

9

- **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila do banco



Introdução

10

- ❑ **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila

ATENDIMENTO



Introdução

11

- ❑ **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila de

ATENDIMENTO



Introdução

12

- ❑ **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila do banco



Introdução

13

- ❑ **Ordem de entrada x Ordem de saída**
 - É comum: ordem de entrada → saída
 - Exemplo: fila do banco



Introdução

14

❑ Resumindo..

- O primeiro a chegar...
- Será o primeiro a ser atendido

FIFO:
First In
First Out



15

Fila de dados

Estrutura de fila de dados

Aplicações de fila de dados

Fila de dados

16

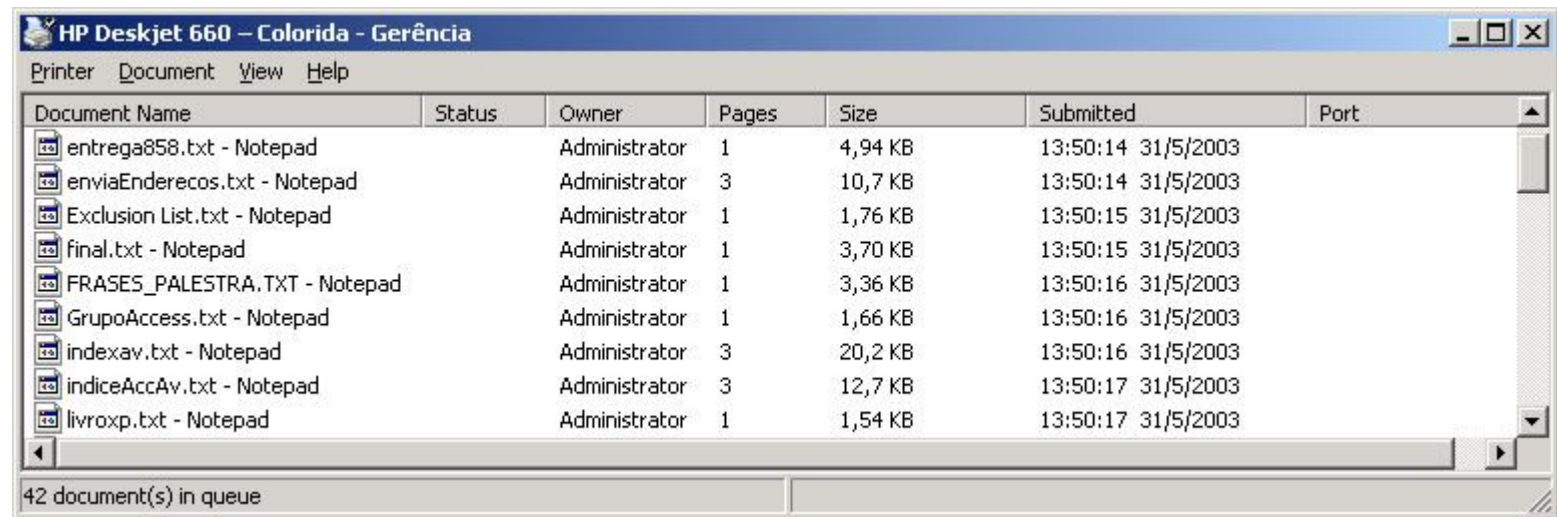
- ❑ Filas de dados
 - Estrutura de dados Fila: Lista FIFO
 - Inserir: sempre no fim da lista (fim da fila)
 - Remover: sempre no início da lista (início da fila)

- ❑ Isso é útil em software?
 - Sim, em muitos casos!

Fila de dados

17

❑ Fila de impressão



The screenshot shows a window titled "HP Deskjet 660 - Colorida - Gerência". It has a menu bar with "Printer", "Document", "View", and "Help". Below the menu is a table listing documents in the print queue. The table has columns for Document Name, Status, Owner, Pages, Size, Submitted, and Port. There are 10 rows of data, all with "Administrator" as the owner and "31/5/2003" as the submission date. At the bottom of the window, it says "42 document(s) in queue".

Document Name	Status	Owner	Pages	Size	Submitted	Port
entrega858.txt - Notepad		Administrator	1	4,94 KB	13:50:14 31/5/2003	
enviaEnderecos.txt - Notepad		Administrator	3	10,7 KB	13:50:14 31/5/2003	
Exclusion List.txt - Notepad		Administrator	1	1,76 KB	13:50:15 31/5/2003	
final.txt - Notepad		Administrator	1	3,70 KB	13:50:15 31/5/2003	
FRASES_PALESTRA.TXT - Notepad		Administrator	1	3,36 KB	13:50:16 31/5/2003	
GrupoAccess.txt - Notepad		Administrator	1	1,66 KB	13:50:16 31/5/2003	
indexav.txt - Notepad		Administrator	3	20,2 KB	13:50:16 31/5/2003	
indiceAccAv.txt - Notepad		Administrator	3	12,7 KB	13:50:17 31/5/2003	
livroxp.txt - Notepad		Administrator	1	1,54 KB	13:50:17 31/5/2003	

42 document(s) in queue

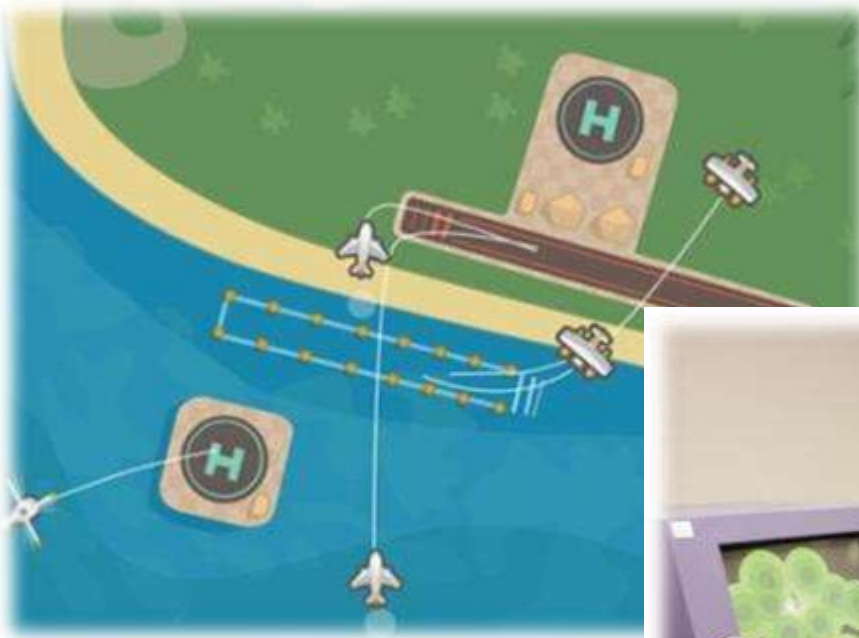
❑ Qual operação faz?

- O sistema operacional coloca os documentos em uma fila!

Fila de dados

18

- Fila de aviões para pouso ou decolagem



Controladores de
tráfegos aéreo

Implementando uma fila estática

Operações: inicializar, fila cheia e vazia, insere e remove

Fila estática

20

- ❑ Na implementação de filas estáticas pode surgir várias formas de construção do tipo de dado fila
 - Vetor e fim separados
 - Usando uma estrutura que conterá o vetor (armazena os elementos da fila) e o fim

```
#define TAM 10
typedef struct fila Fila;

struct fila{
    int info[TAM];
    int fim;
};
```

Fila estática

21

- ❑ Novo elemento é inserido no fim e acesso é apenas ao início
 - O primeiro que entra é o primeiro que sai (FIFO – first in first out)
- ❑ Operações básicas:
 - **Inicializar:** recebe uma fila e aponta seu fim para -1
 - **Inserir:** insere um novo elemento no fim
 - **Remove:** remove um elemento do início
 - **Fila vazia:** verifica se fila está vazia
 - **Fila cheia:** verificar se fila está cheia

Implementando uma fila estática

22

❑ Elementos da fila: inseridos em um vetor

Fila:	0	1	2	3	4	5	6	7	8	9
	?	?	?	?	?	?	?	?	?	?

Fim: ??

❑ Operações:

- Inicializar
- Fila cheia e vazia
- Insere
- Remove

Implementando uma fila estática

23

❑ Inicializar uma fila

Fila:

0	1	2	3	4	5	6	7	8	9
?	?	?	?	?	?	?	?	?	?

Fim: -1



❑ Recebe uma fila e aponta seu fim para -1

- Fim sempre indica o último elemento! Inicializar o fim com valor negativo indica que a fila está vazia

Implementando uma fila estática

24

❑ Insere

Fila:	0	1	2	3	4	5	6	7	8	9
	?	?	?	?	?	?	?	?	?	?

Fim: -1

❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1)$ → Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 8?

Implementando uma fila estática

25

❑ Insere

Fila:

0	1	2	3	4	5	6	7	8	9
?	?	?	?	?	?	?	?	?	?

Fim: -1

-1 < 9...
Posso inserir!

❑ Como inserir um valor?

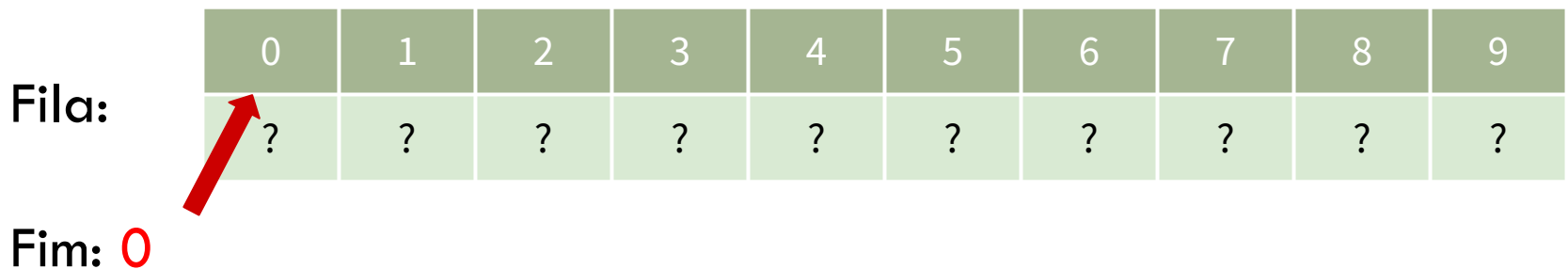
- Se $\text{fim} < (\text{TAM} - 1) \rightarrow$ Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 8?

Implementando uma fila estática

26

❑ Insere



❑ Como inserir um valor?

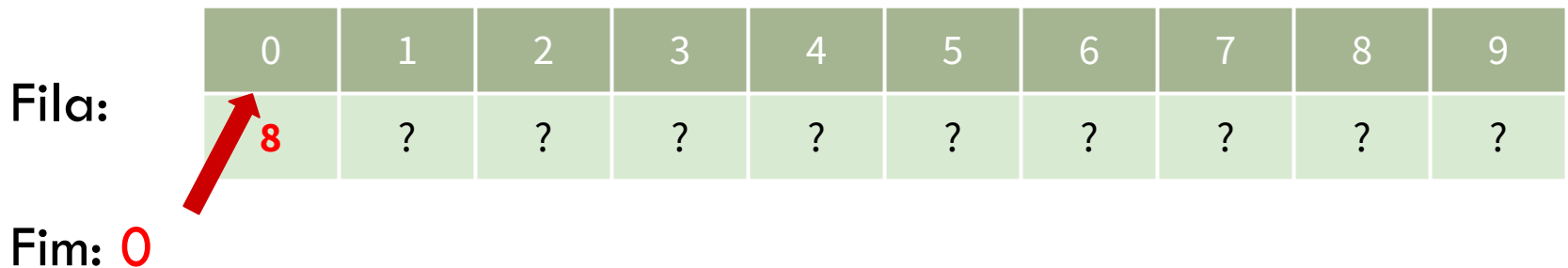
- Se $\text{fim} < (\text{TAM} - 1) \rightarrow$ Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 8?

Implementando uma fila estática

27

❑ Insere



❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1) \rightarrow$ Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 8?

Implementando uma fila estática

28

❑ Inserir

Fila:	0	1	2	3	4	5	6	7	8	9
	8	5	10	18	45	19	29	55	60	?

Fim: 8

❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1)$ → Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 78?

Implementando uma fila estática

29

❑ Inserir

Fila:

0	1	2	3	4	5	6	7	8	9
8	5	10	18	45	19	29	55	60	?

Fim: 8

8 < 9...
Posso inserir!

❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1)$ → Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 78?

Implementando uma fila estática

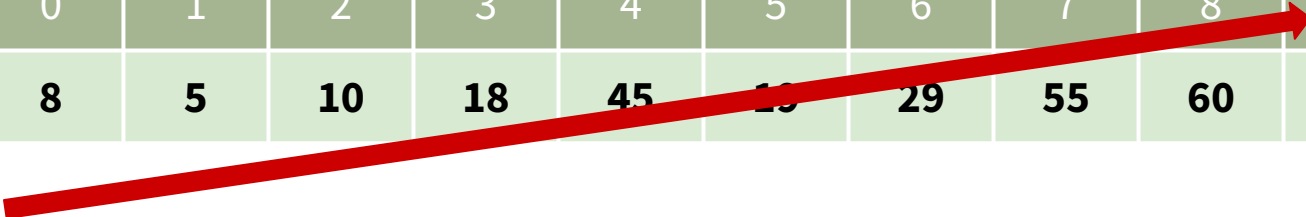
30

❑ Inserir

Fila:

0	1	2	3	4	5	6	7	8	9
8	5	10	18	45	19	29	55	60	?

Fim: 9



❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1) \rightarrow$ Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número **78**?

Implementando uma fila estática

31

❑ Inserir

Fila:	0	1	2	3	4	5	6	7	8	9
	8	5	10	18	45	19	29	55	60	78

Fim: 9

❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1)$ → Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 78?

Implementando uma fila estática

32

❑ Inserir

Fila:	0	1	2	3	4	5	6	7	8	9
	8	5	10	18	45	19	29	55	60	78

Fim: 9

❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1)$ → Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 20?

Implementando uma fila estática

33

❑ Inserir

Fila:

0	1	2	3	4	5	6	7	8	9
8	5	10	18	45	19	29	55	60	78

Fim: 9

9 < 9? Não!

❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1)$ → Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 20?

Implementando uma fila estática

34

❑ Inserir

Fila:

0	1	2	3	4	5	6	7	8	9
8	5	10	18	45	19	29	55	60	78

Fim: 9

Fila cheia!

❑ Como inserir um valor?

- Se $\text{fim} < (\text{TAM} - 1)$ → Pode inserir
- Soma 1 no fim... E acrescenta-se elemento na posição do vetor

❑ Vamos inserir o número 20?

Implementando uma fila estática

35

❑ Inicializar

- Recebe uma fila e aponta seu fim para -1

```
void fila_inicializa (Fila* f){  
    f->fim = -1;  
}
```

Implementando uma fila estática

36

❑ Fila cheia

- Verifica se a fila está cheia, caso esteja, não pode inserir

```
int fila_cheia(Fila* f){  
    if(f->fim >= TAM - 1)  
        return 1;  
    return 0;  
}
```

Fila cheia!



Implementando uma fila estática

37

❑ Insere

- Se a fila não estiver cheia, incrementa o fim e insere um novo elemento na posição do fim atual

```
int fila_insere(Fila* f, int valor){  
    if(!fila_cheia(f)){  
        f->fim++;  
        f->info[f->fim] = valor;  
        return 1;  
    }  
    return 0;  
}
```

Implementando uma fila estática

38

❑ Remove

Fila:	0	1	2	3	4	5	6	7	8	9
	8	5	10	18	45	19	?	?	?	?

Fim: 5

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover um número?

Implementando uma fila estática

39

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
8	5	10	18	45	19	?	?	?	?

Fim: 5

5 != -1 ...
Posso remover!

❑ Como remover um valor?

- Se o fim != -1 → Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover um número?

Implementando uma fila estática

40

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
8	5	10	18	45	19	?	?	?	?

Fim: 5



Removemos:
8

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover um número?

Implementando uma fila estática


41

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
8	5	10	18	45	19	?	?	?	?

Fim: 5



Deslocamos os
elementos!

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover um número?

Implementando uma fila estática


42

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
5	10	18	45	19	19	?	?	?	?

Fim: 5



Deslocamos os
elementos!

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover um número?

Implementando uma fila estática

43

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
5	10	18	45	19	19	?	?	?	?

Fim: 4

Subtrai 1 do fim!

❑ Como remover um valor?

- Se o fim $\neq -1$ → Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover um número?

Implementando uma fila estática

44

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
5	10	18	45	19	19	?	?	?	?

Fim: 4

❑ Como remover um valor?

- Se o fim $\neq -1$ → Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática

45

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
5	10	18	45	19	19	?	?	?	?

Fim: 4

4 != -1 ...
Posso remover!

❑ Como remover um valor?

- Se o fim != -1 → Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática


46

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
5	10	18	45	19	19	?	?	?	?

Fim: 4



Removemos:
5

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática


47

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
5	10	18	45	19	19	?	?	?	?

Fim: 4



Deslocamos os
elementos!

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática

48

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
10	18	45	19	19	19	?	?	?	?

Fim: 4

Deslocamos os
elementos!

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática

49

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
10	18	45	19	19	19	?	?	?	?

Fim: 3

Subtrai 1 do fim!

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática

50

❑ Remove

Fila:	0	1	2	3	4	5	6	7	8	9
	19	19	19	19	19	19	?	?	?	?

Fim: -1

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática

51

❑ Remove

Fila:	0	1	2	3	4	5	6	7	8	9
	19	19	19	19	19	19	?	?	?	?

Fim: **-1**

-1 != -1? Não!

❑ Como remover um valor?

- Se o fim != -1 → Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática

52

❑ Remove

Fila:

0	1	2	3	4	5	6	7	8	9
19	19	19	19	19	19	?	?	?	?

Fim: -1

Fila vazia!

❑ Como remover um valor?

- Se o fim $\neq -1 \rightarrow$ Pode remover
- Lê o elemento...
- Desloca os elementos para o início da fila
- E subtrai 1 do fim

❑ Vamos remover outro número?

Implementando uma fila estática

53

❑ Fila vazia

- Verifica se a fila está vazia, caso esteja, não pode remover

```
int fila_vazia(Fila* f){  
    if(f->fim == -1)  
        return 1;  
    return 0;  
}
```

Fila vazia!



Implementando uma fila estática

54

❑ Remove

- Se a fila não estiver vazia, lê o elemento, desloca os elementos para o início da fila e subtrai 1 do fim

```
int fila_remove(Fila *f){
    int valor, i;
    if(!fila_vazia(f)){
        valor = f->info[0];
        for (i = 0 ; i < f->fim; i++){
            f->info[i] = f->info[i+1];
        }
        f->fim--;
    }else{
        exit(1);
    }
    return valor;
}
```

Implementando uma fila estática

Atividade

55

❑ Imprimir?

- Imprimir os elementos da fila sem usar o remove, pois o objetivo é só observar o conteúdo da fila sem modificar o seu conteúdo

Implementando uma fila dinâmica

Operações: inicializar, fila cheia e vazia, insere e remove

Fila dinâmica

57

- ❑ Implementar uma fila dinâmica usando uma lista encadeada simples

```
typedef struct filaD FilaD;  
  
struct filaD{  
    int info;  
    FilaD* prox;  
};
```

Fila dinâmica

58

- ❑ Novo elemento é inserido no fim e acesso é apenas ao início
 - O primeiro que entra é o primeiro que sai (FIFO – first in first out)

- ❑ Operações básicas:
 - **Inicializar:** cria uma fila vazia, representada pelo ponteiro NULL
 - **AlocaNo:** aloca memória para armazenar o elemento (nó)
 - **Inserir:** insere um novo elemento no fim
 - **Remove:** remove um elemento do início da Fila
 - **Vazia:** verifica se fila está vazia

Implementando uma fila dinâmica

59

❏ Inicializar

- Cria uma fila vazia, representada pelo ponteiro NULL

```
FilaD* filaD_inicializa (){\n    return NULL;\n}
```

Fila → NULL

Implementando uma fila dinâmica

60

❏ Insere

- Aloca memória para armazenar o elemento
- Encadeia o elemento na fila existente

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```

Implementando uma fila dinâmica

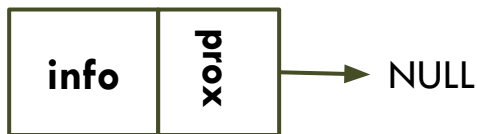
61

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```

alocaNo



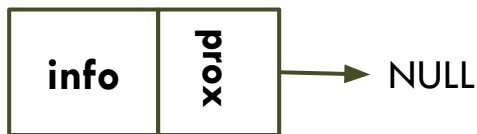
Implementando uma fila dinâmica

62

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



Implementando uma fila dinâmica

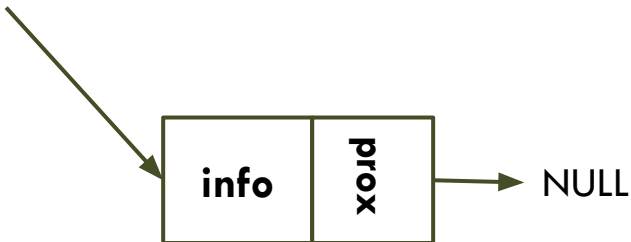
63

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```

Fila



Implementando uma fila dinâmica

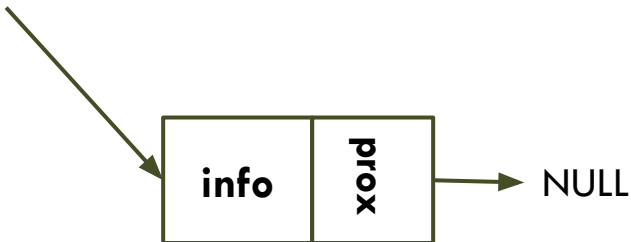
64

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```

Fila



Implementando uma fila dinâmica

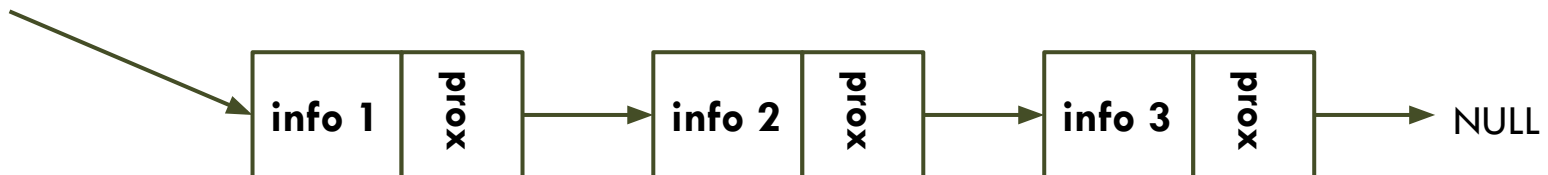
65

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```

Fila



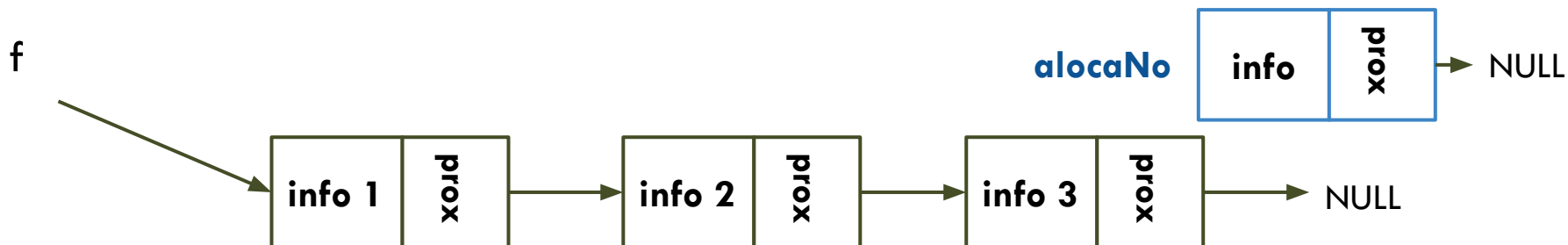
Implementando uma fila dinâmica

66

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



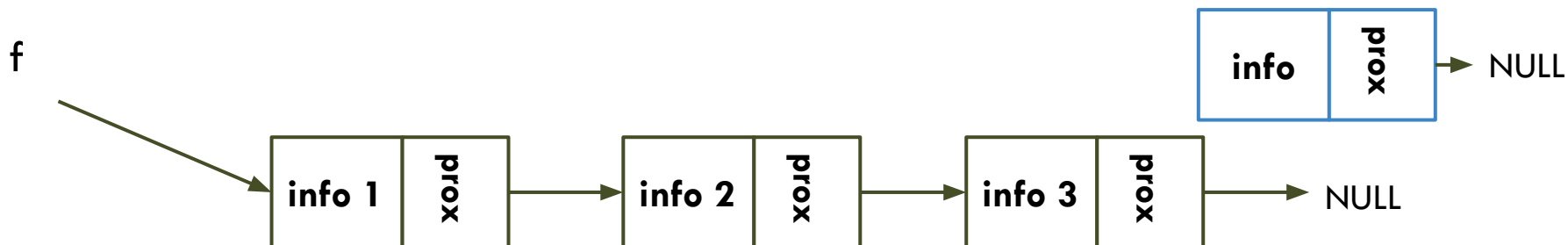
Implementando uma fila dinâmica

67

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



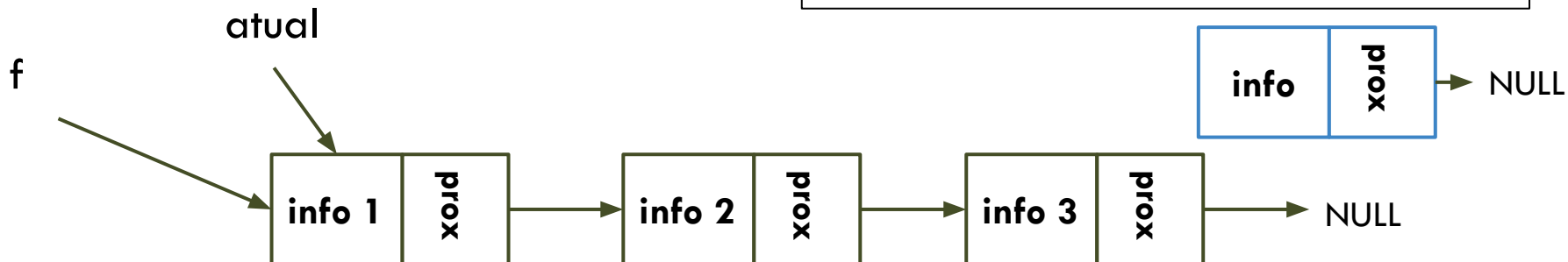
Implementando uma fila dinâmica

68

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



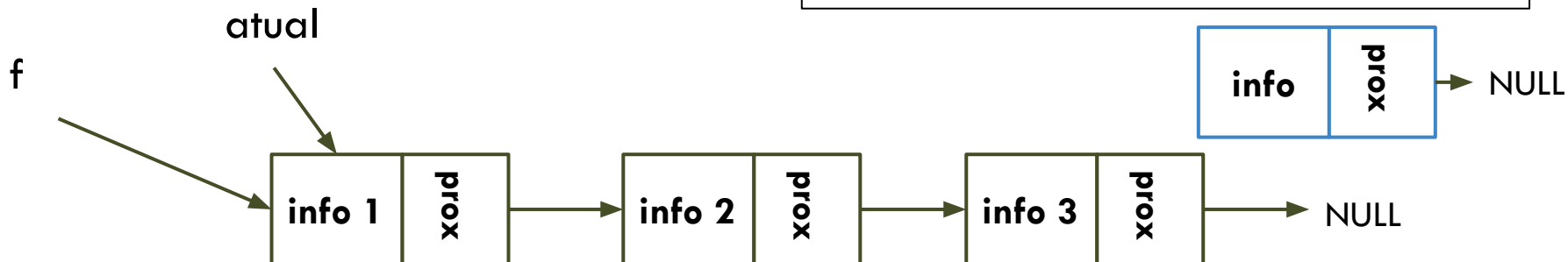
Implementando uma fila dinâmica

69

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



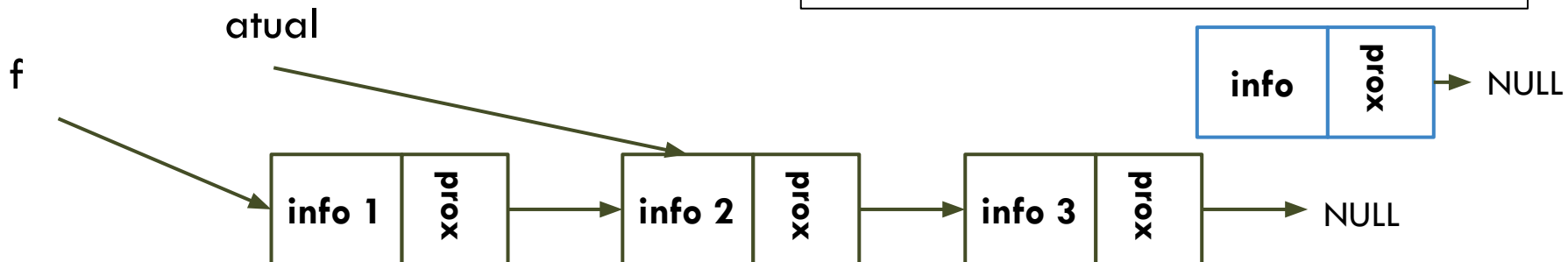
Implementando uma fila dinâmica

70

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



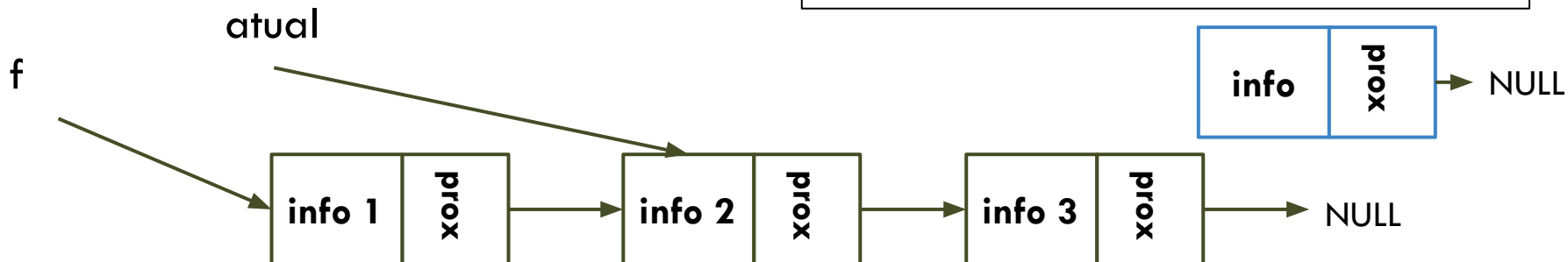
Implementando uma fila dinâmica

71

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



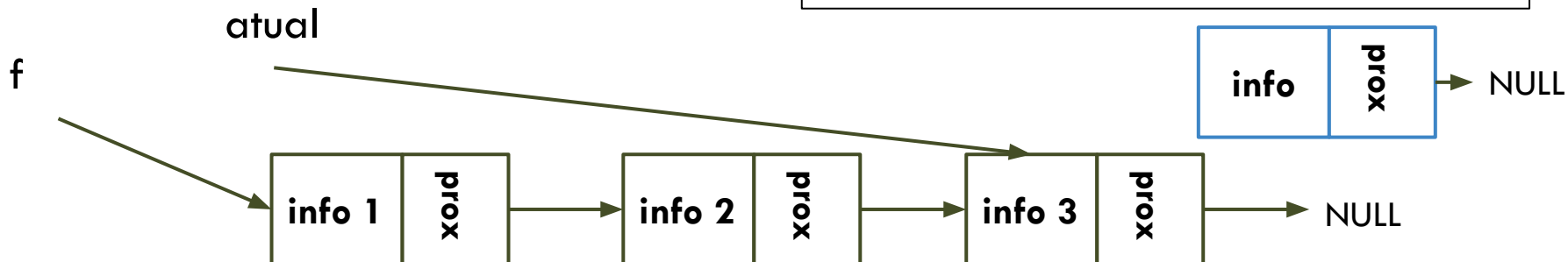
Implementando uma fila dinâmica

72

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



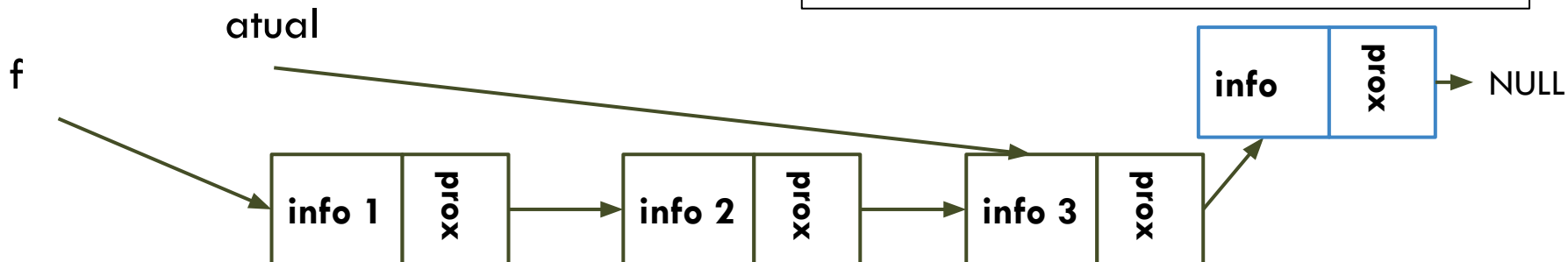
Implementando uma fila dinâmica

73

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```



Implementando uma fila dinâmica

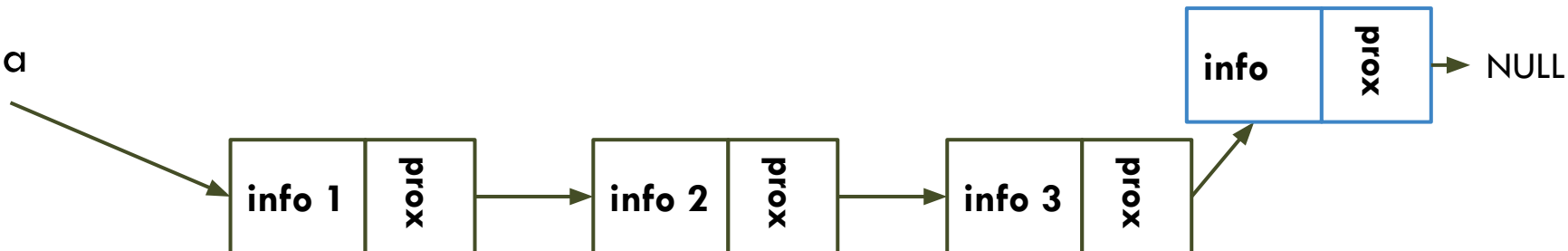
74

❏ Insere

```
FilaD* alocaNo(int valor){  
    FilaD* no = (FilaD*) malloc(sizeof(FilaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
FilaD* filaD_insere(FilaD* f, int valor){  
    FilaD *novo = alocaNo(valor);  
    if (filaD_vazia(f)){  
        f = novo;  
    }else{  
        FilaD* atual = f;  
        while (atual->prox != NULL){  
            atual = atual->prox;  
        }  
        atual->prox = novo;  
    }  
    return f;  
}
```

Fila



Implementando uma fila dinâmica

75

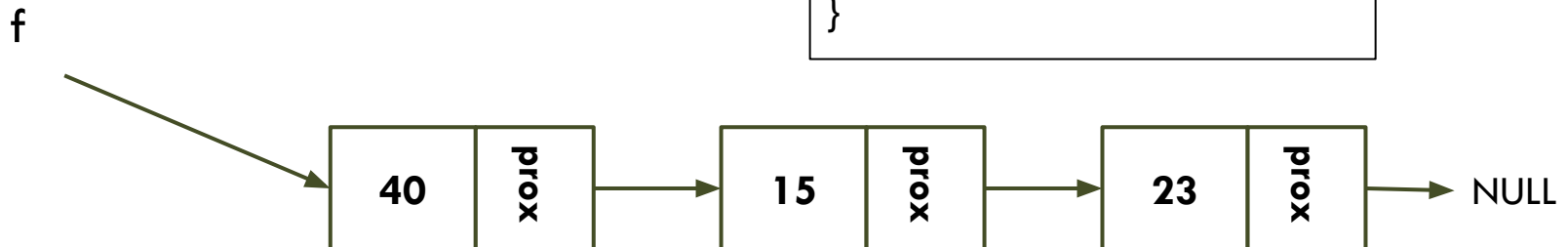
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
remove(&f);  
remove(&f);  
remove(&f);  
remove(&f);
```



Implementando uma fila dinâmica

76

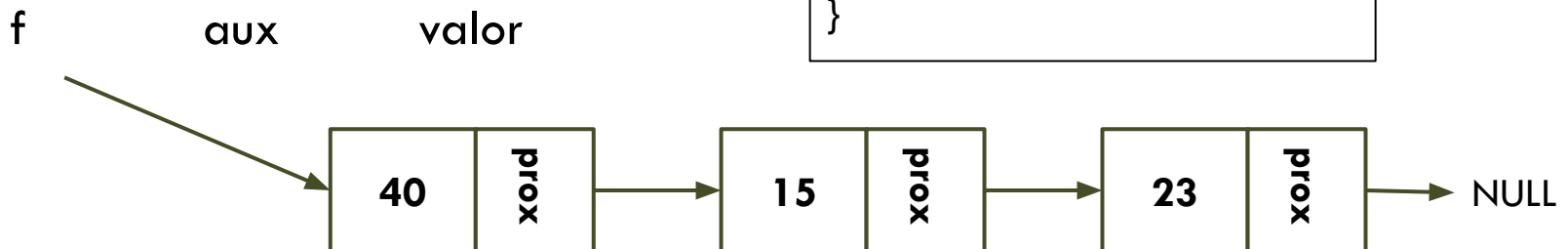
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
remove(&f);  
remove(&f);  
remove(&f);  
remove(&f);
```



Implementando uma fila dinâmica

77

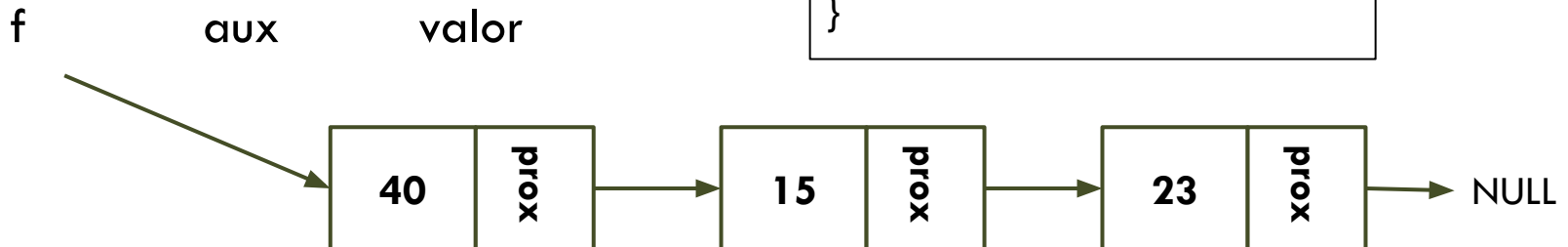
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

78

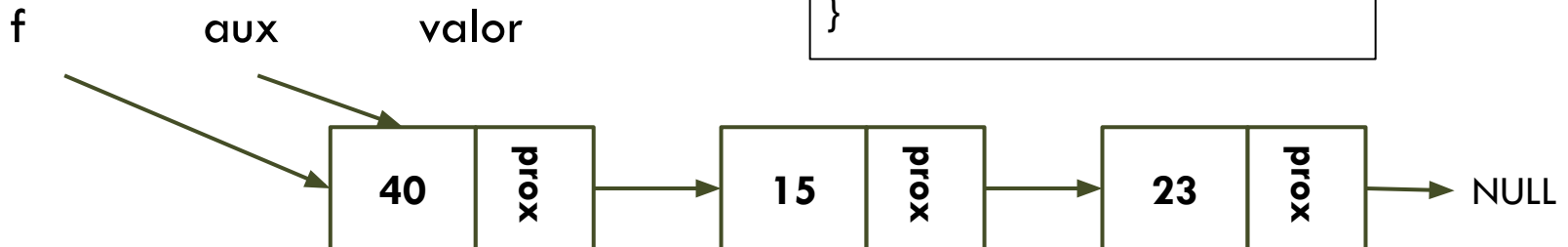
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
remove(&f);  
remove(&f);  
remove(&f);  
remove(&f);
```



Implementando uma fila dinâmica

79

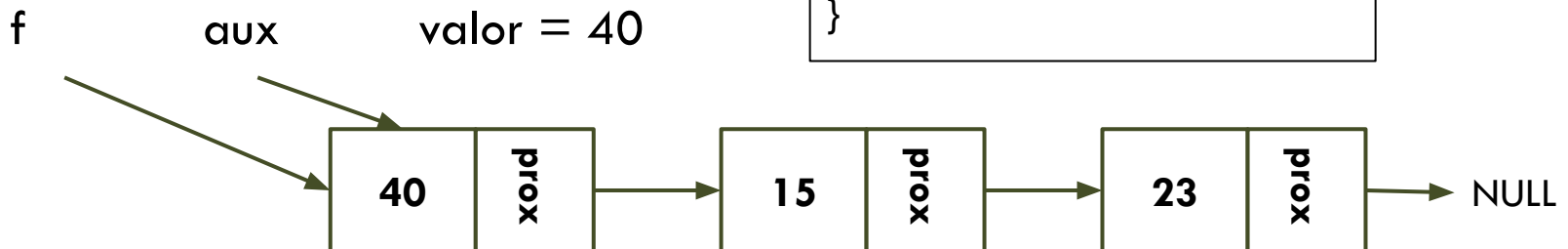
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

80

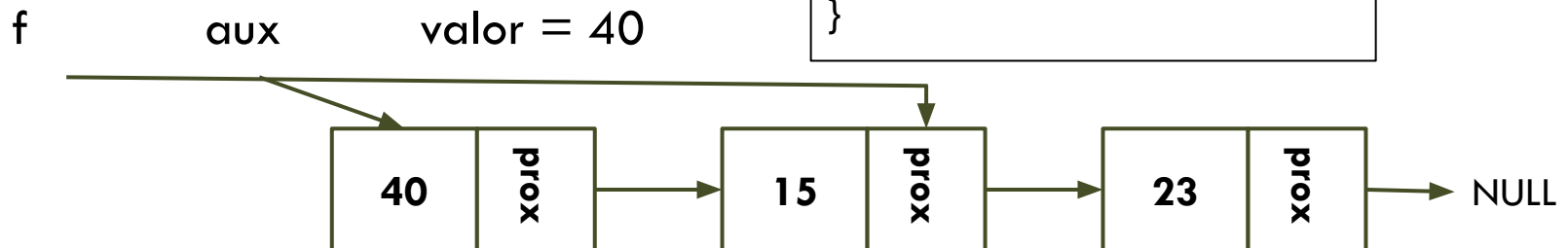
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

81

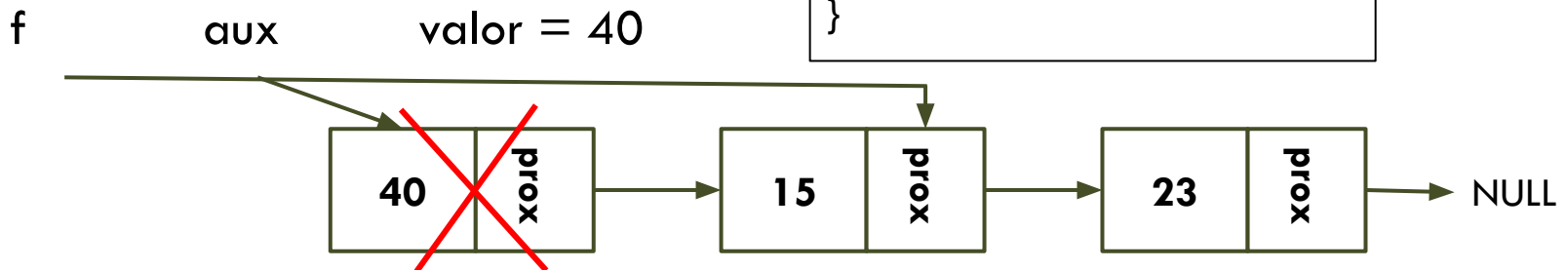
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

82

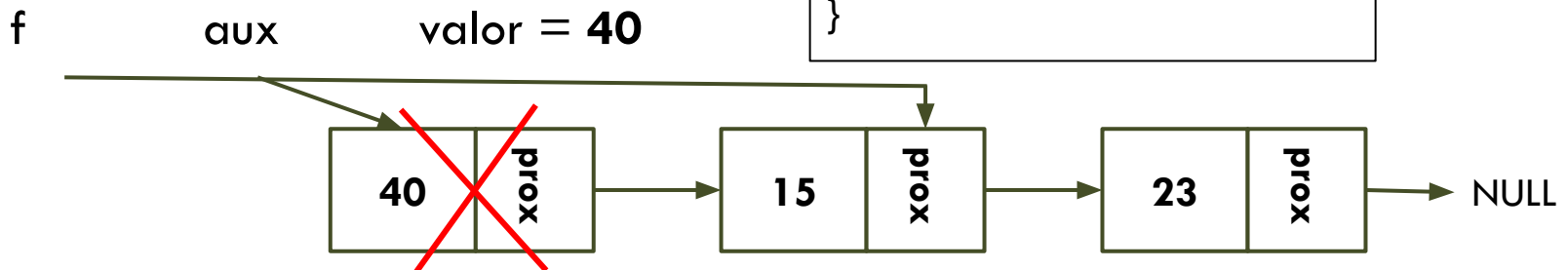
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

83

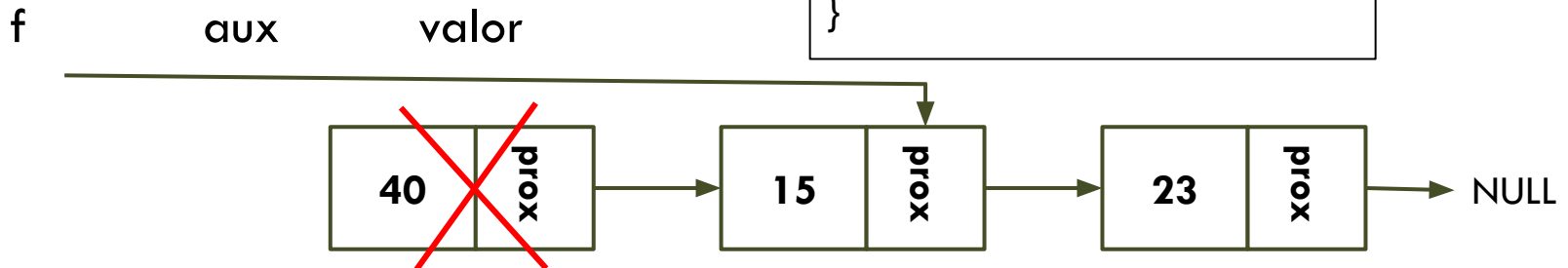
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

84

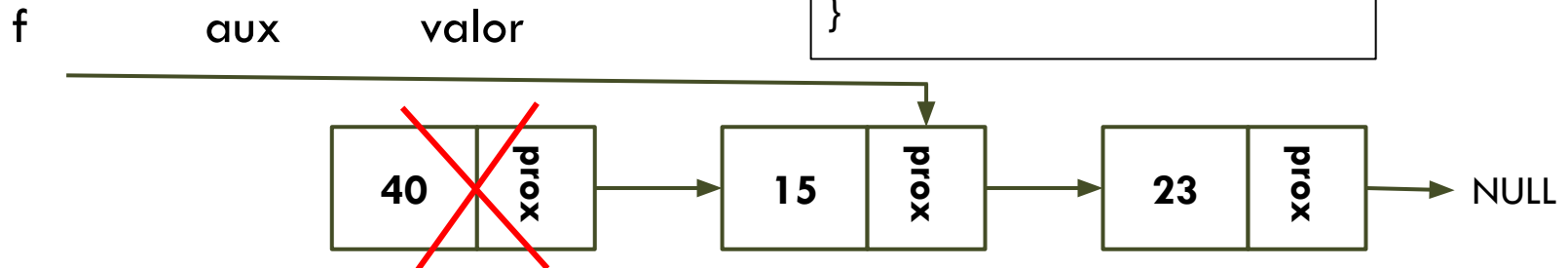
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

85

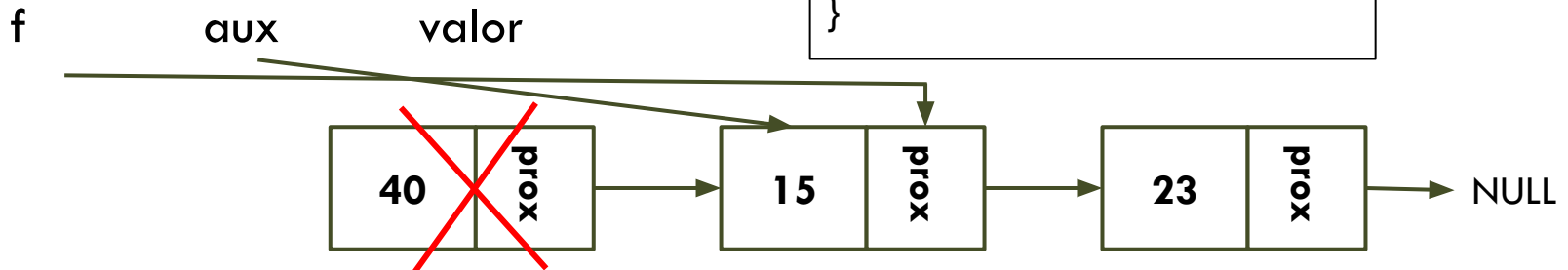
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

86

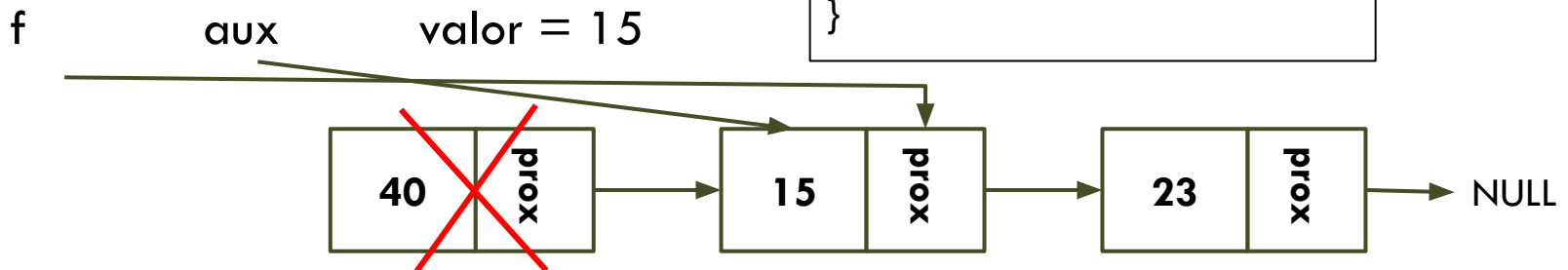
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

87

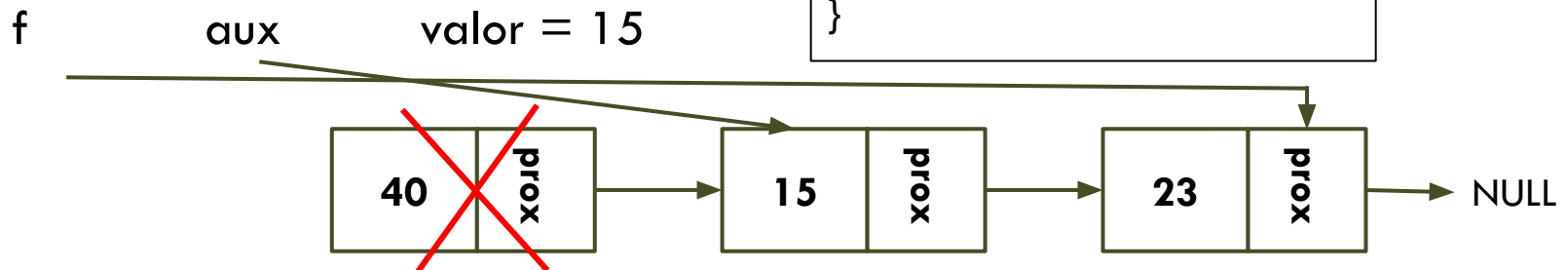
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

88

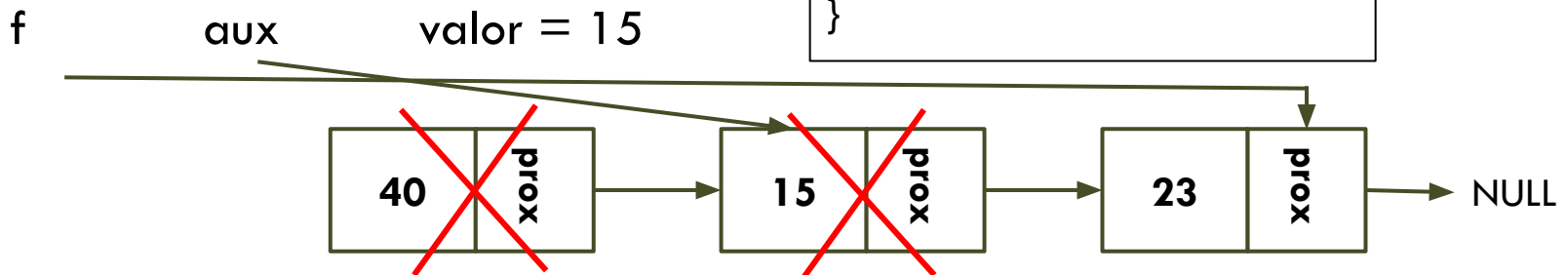
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

89

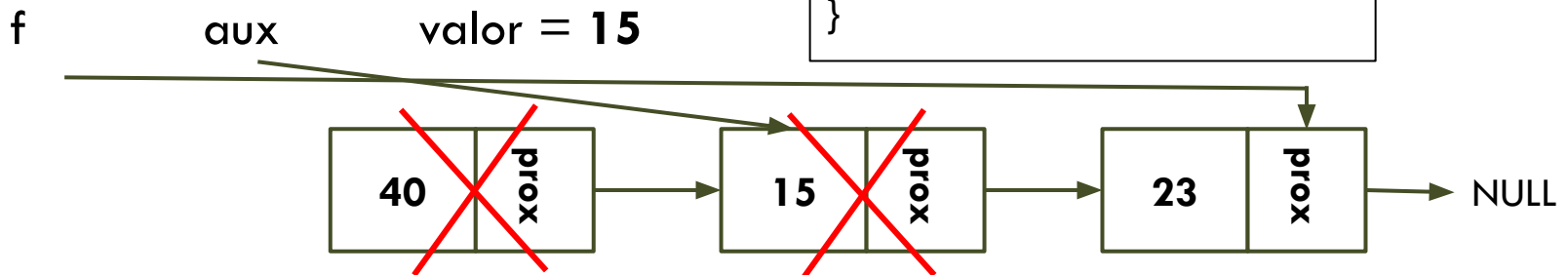
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

90

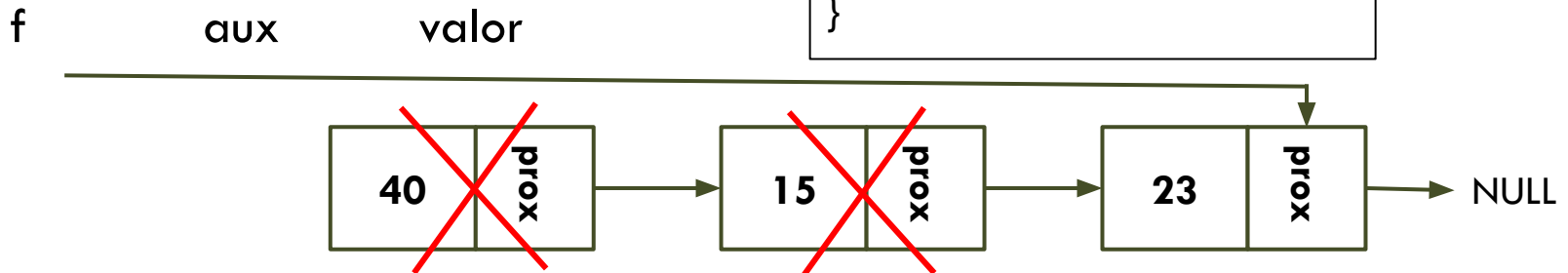
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

91

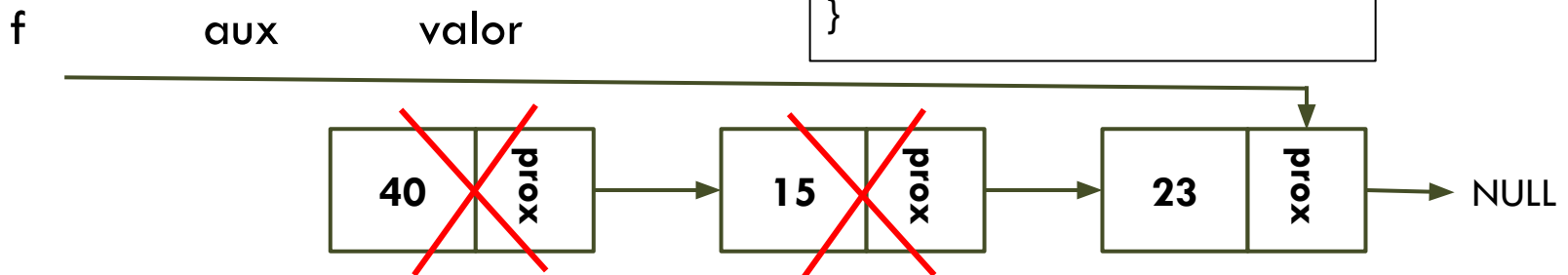
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

**remove(&f);
remove(&f);
remove(&f);
remove(&f);**



Implementando uma fila dinâmica

92

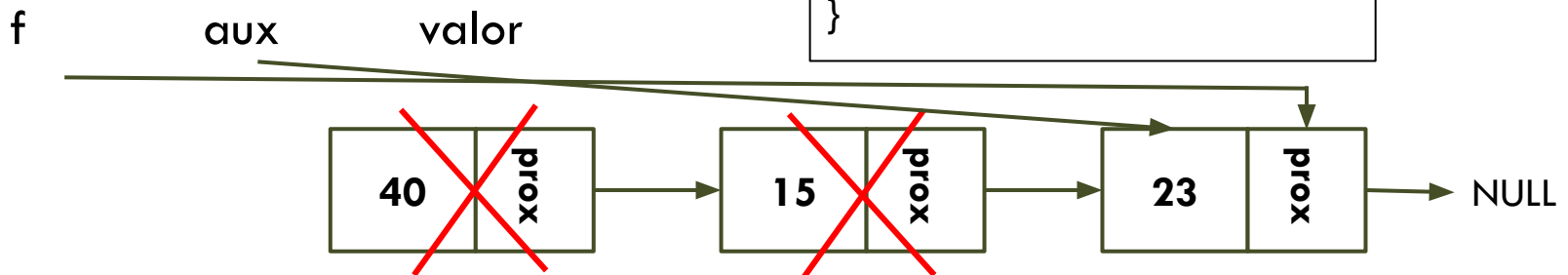
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

93

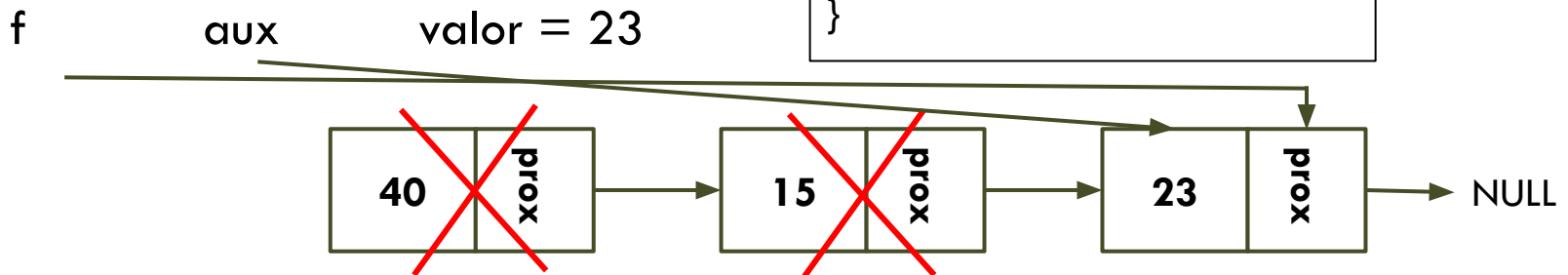
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

94

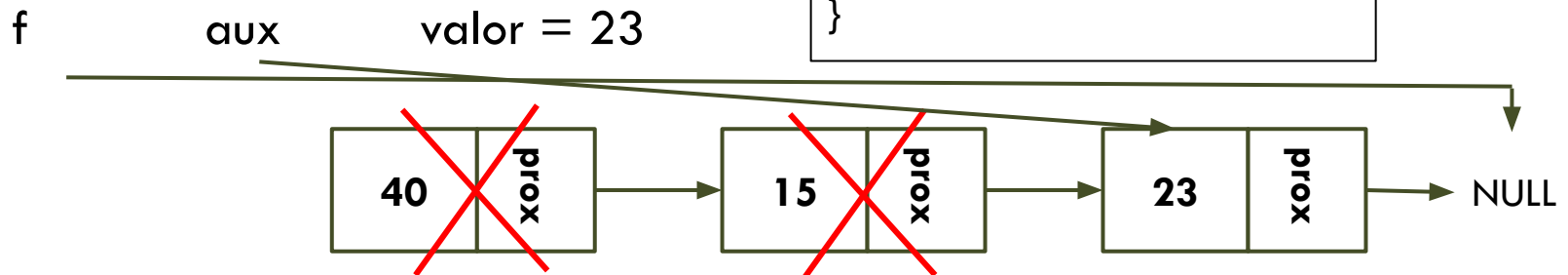
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

95

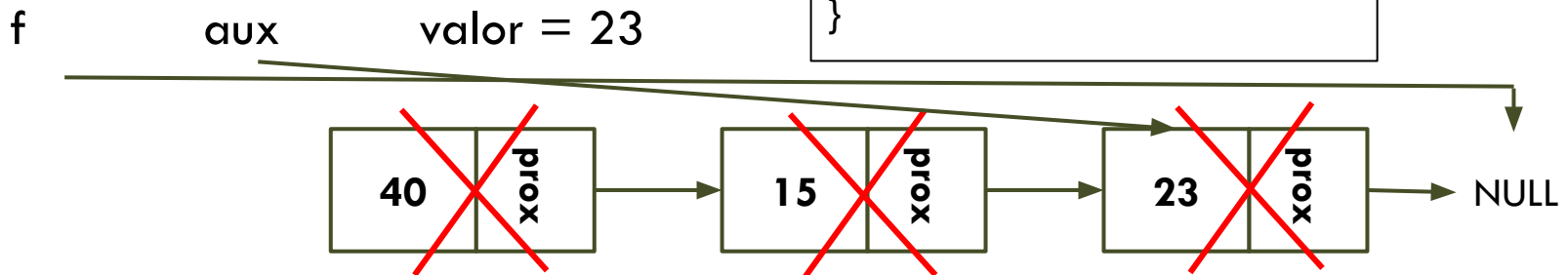
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

96

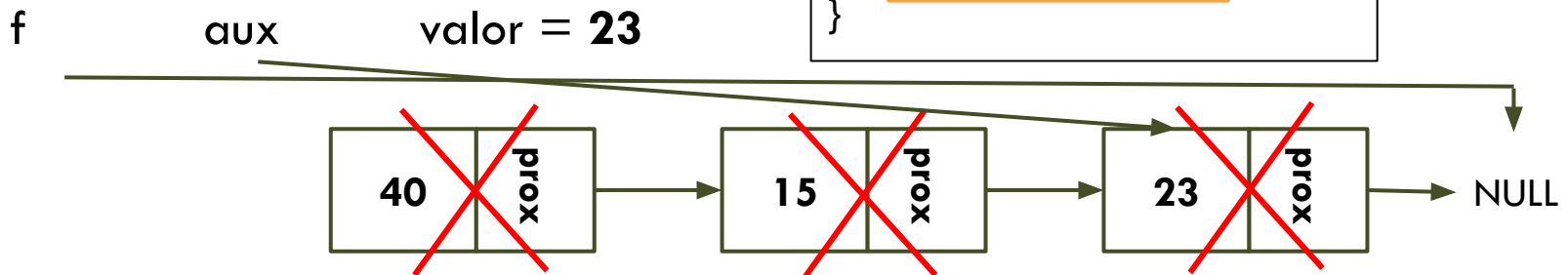
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);



Implementando uma fila dinâmica

97

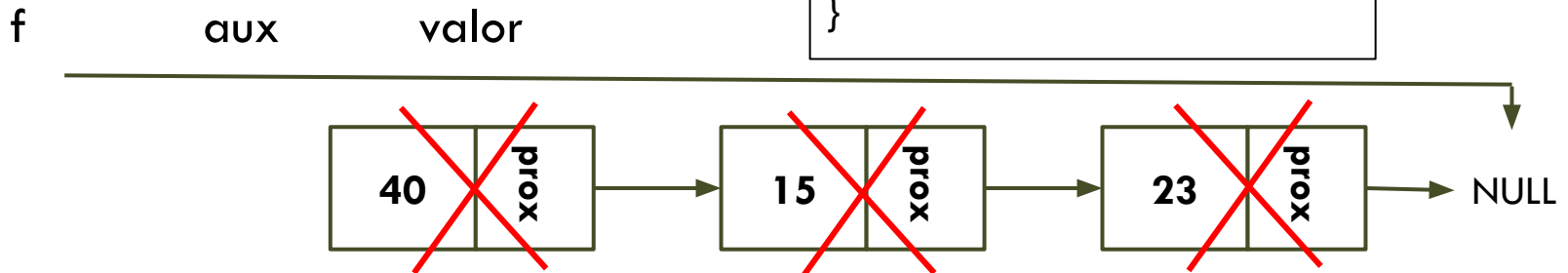
Remove

- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

**remove(&f);
remove(&f);
remove(&f);
remove(&f);**



Implementando uma fila dinâmica

98

Remove

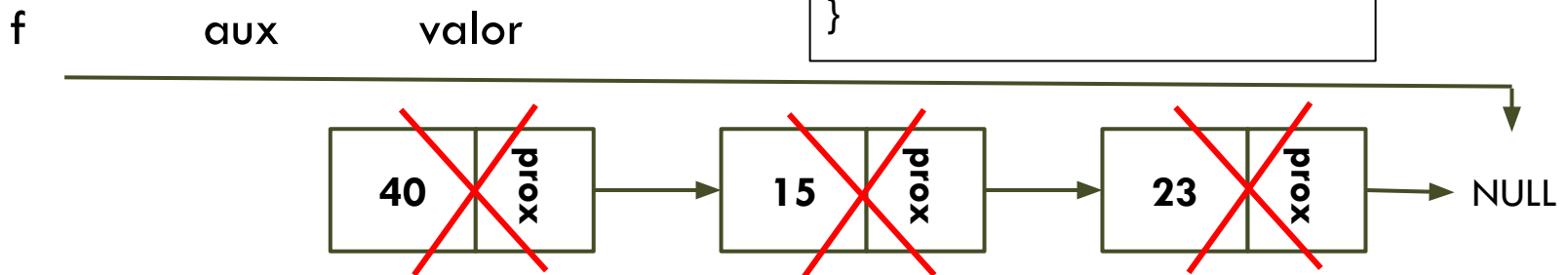
- Retira o elemento do início da lista

```
int filaD_vazia(FilaD* f){  
    if(f == NULL)  
        return 1;  
    return 0;  
}
```

```
int filaD_remove(FilaD** f){  
    FilaD* aux;  
    int valor;  
    if (!filaD_vazia(*f)){  
        aux = *f;  
        valor = (*f)->info;  
        *f = (*f)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

remove(&f);
remove(&f);
remove(&f);
remove(&f);

Fila vazia!



Implementando uma fila dinâmica

Atividade

99

❑ **Imprimir?**

- Imprimir os elementos da fila sem usar o remove, pois o objetivo é só observar o conteúdo da fila sem modificar o seu conteúdo

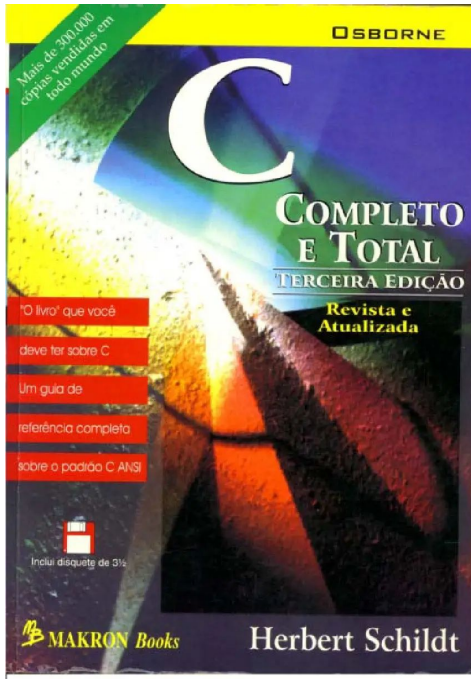
❑ **Libera?**

- Libera a fila depois de liberar todos os elementos (nó) alocados

Referências



100



SCHILD, Herbert. **C completo e total**. Makron, 3ª edição revista e atualizada, 1997.



SZWARCHFITER, J. **Estruturas de Dados e seus algoritmos**. 3 ed. Rio de Janeiro: LTC, 2010.