

FILA, PILHA E DEQUE

Prof. Muriel Mazzetto
Estrutura de Dados

Listas

2

- Estruturas derivadas de listas:
 - ▣ Listas circulares;
 - ▣ Com nó descritor;
 - ▣ Encadeada Dinâmica;
 - ▣ Duplamente Encadeada Dinâmica;
 - ▣ Sequencial Estática.

Listas

3

- Estruturas derivadas de listas:
 - ▣ Listas circulares;
 - ▣ Com nó descritor;
 - ▣ Encadeada Dinâmica;
 - ▣ Duplamente Encadeada Dinâmica;
 - ▣ Sequencial Estática.
- Todas possibilitam acessar qualquer elemento de dentro da lista (buscas, inserções ordenadas, remoção de elementos internos).

Listas

4

- Filas, Pilhas e Deques derivam das listas.
- **Restringem o acesso** aos elementos em uma **ordem específica**.
- Impossibilitam acesso aos elementos internos, alterando apenas as **extremidades**.

Fila

5

- Tipo especial de lista.
- Sentido de inserção e remoção.

Fila

6

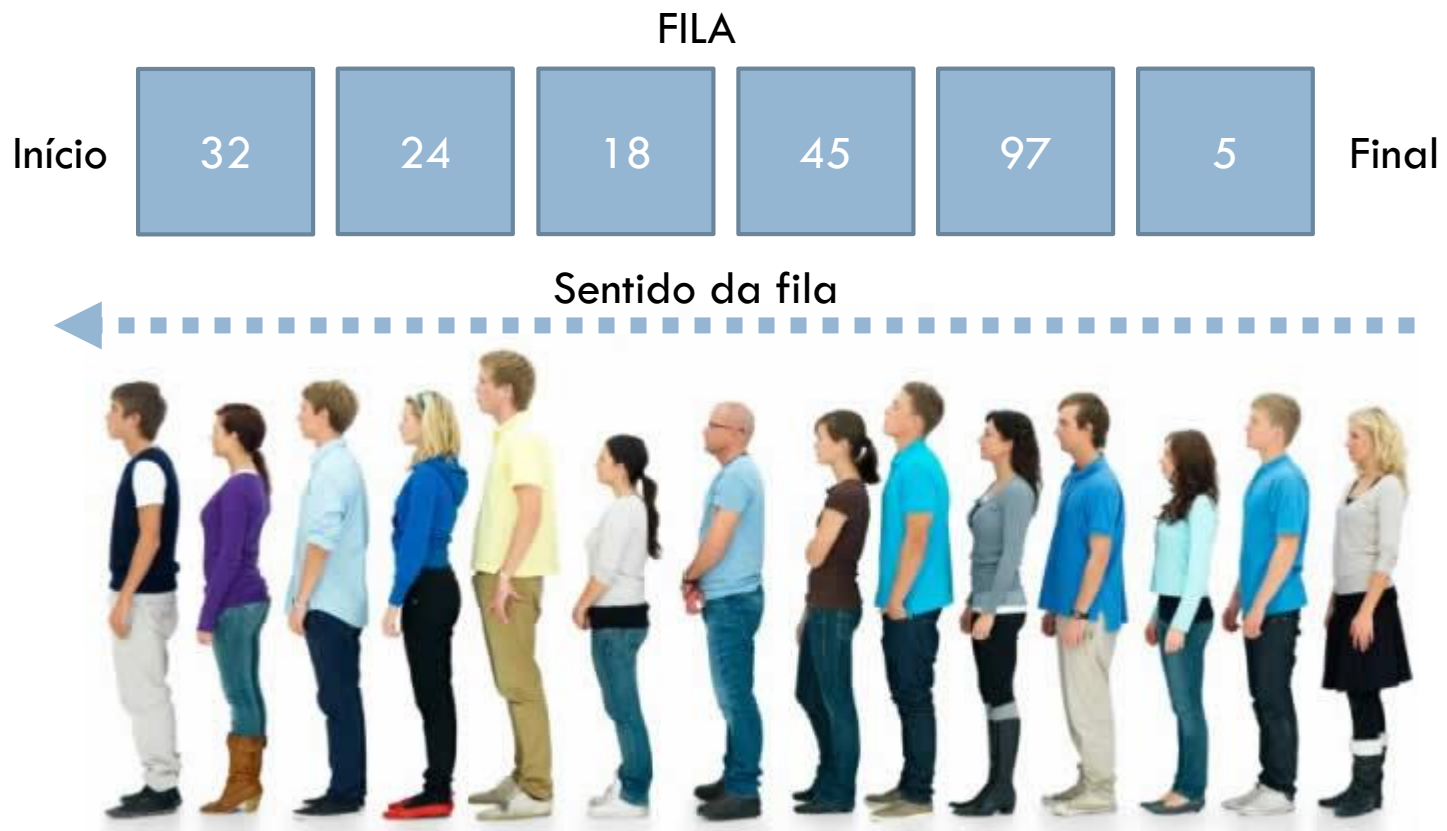
- ❑ Tipo especial de lista.
- ❑ Sentido de inserção e remoção.



Fila

7

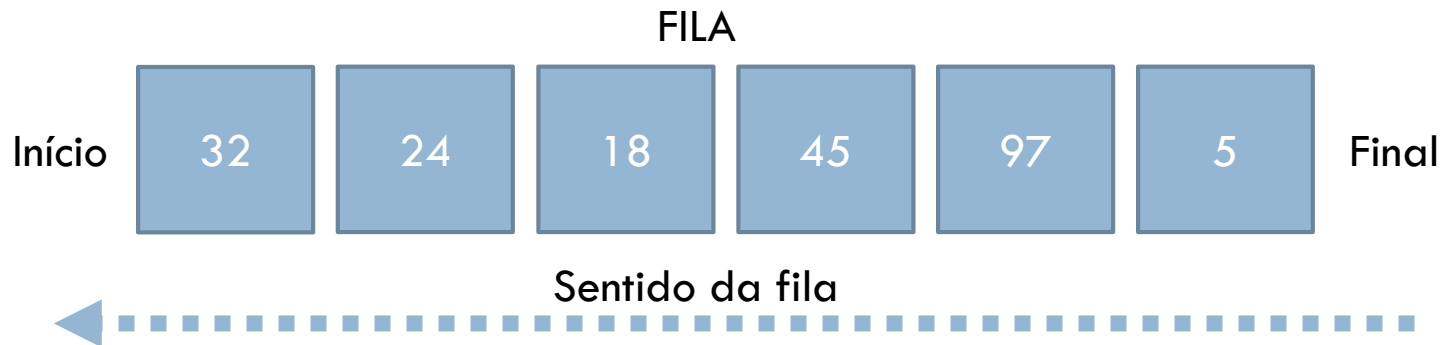
- ❑ Tipo especial de lista.
- ❑ Sentido de inserção e remoção.



Fila

8

- ❑ Tipo especial de lista.
- ❑ Sentido de inserção e remoção.

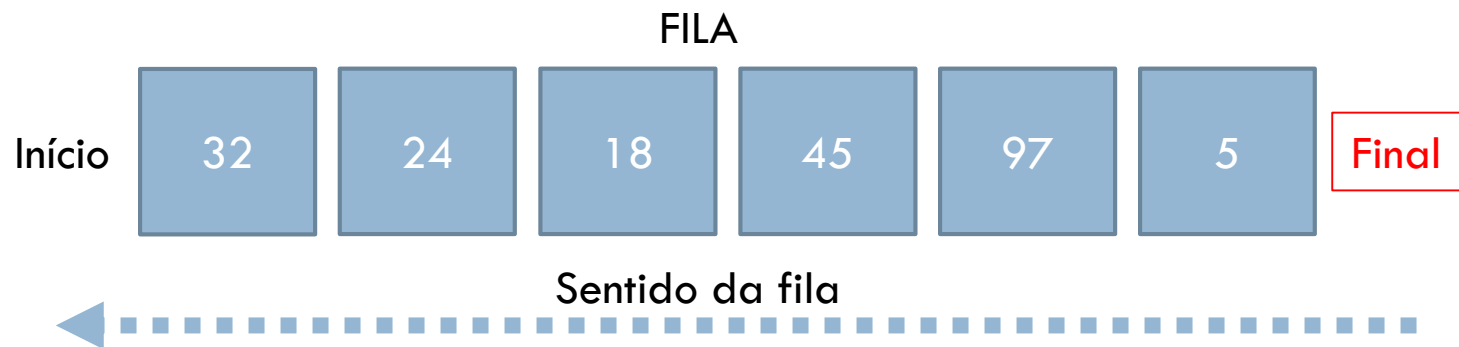


INSERIR ELEMENTOS NO FINAL.
REMOVER ELEMENTOS DO INÍCIO.
FIFO (First In, First Out)

Fila

9

- ❑ Tipo especial de lista.
- ❑ Sentido de inserção e remoção.

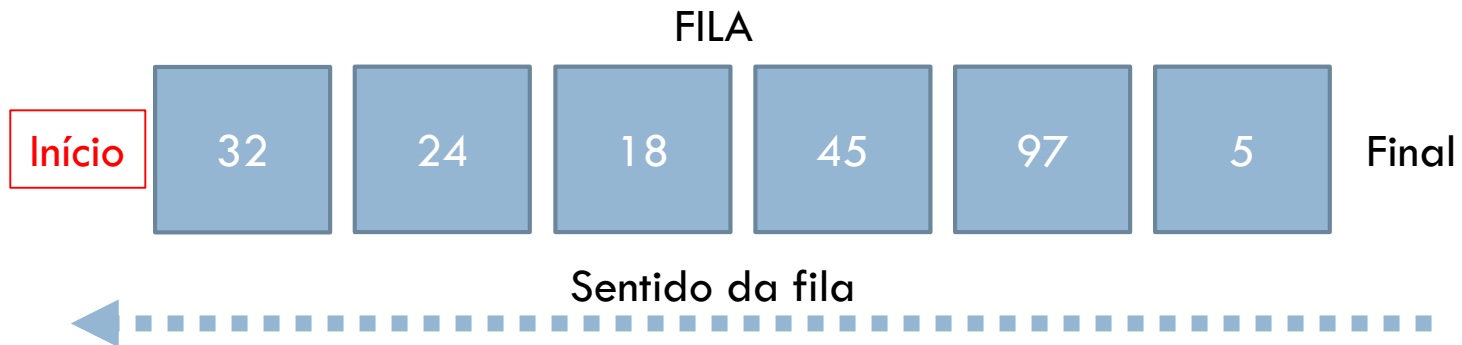


INSERIR ELEMENTOS NO FINAL.
REMOVER ELEMENTOS DO INÍCIO.
FIFO (First In, First Out)

Fila

10

- ❑ Tipo especial de lista.
- ❑ Sentido de inserção e remoção.

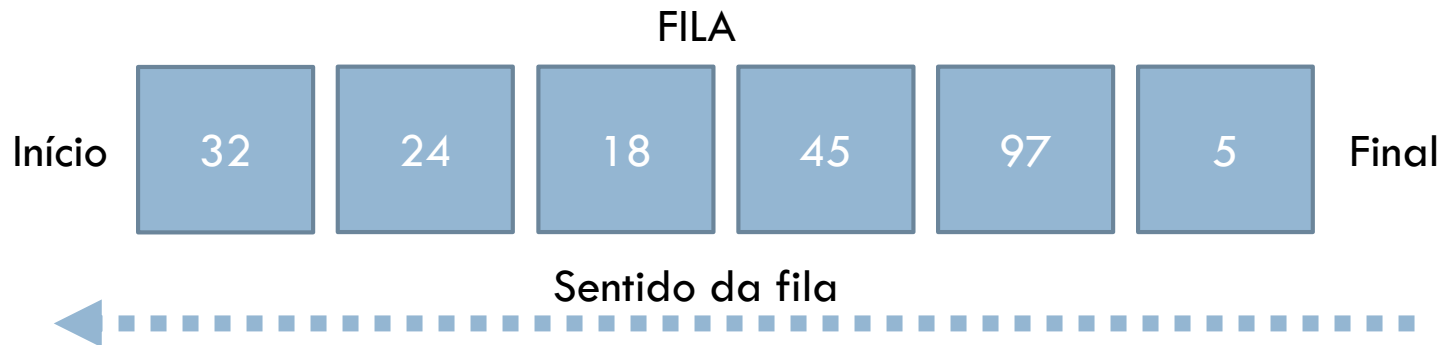


INSERIR ELEMENTOS NO FINAL.
REMOVER ELEMENTOS DO INÍCIO.
FIFO (First In, First Out)

Fila

11

- ❑ Tipo especial de lista.
- ❑ Sentido de inserção e remoção.



INSERIR ELEMENTOS NO FINAL.
REMOVER ELEMENTOS DO INÍCIO.
FIFO (First In, First Out)

Fila

12

- Aplicações:
 - ▣ Controle de fluxo.
 - ▣ Recursos compartilhados.

Fila

13

- Operações comuns de uma Fila:
 - ▣ Criação da fila (Alocar);
 - ▣ Inserção de um elemento no final (ENFILAR);
 - ▣ Remoção de um elemento do início (DESENFILAR);
 - ▣ Acesso ao elemento (Início);
 - ▣ Destruição da fila (Desalocar).

Fila

14

- Operações comuns de uma Fila:
 - ▣ Criação da fila (Alocar);
 - ▣ Inserção de um elemento no final (ENFILAR);
 - ▣ Remoção de um elemento do início (DESENFILAR);
 - ▣ Acesso ao elemento (Início);
 - ▣ Destruição da fila (Desalocar).

- Fila estática: derivada de listas estáticas.
- Fila dinâmica: derivada de listas encadeadas com nó descritor.

Fila Dinâmica: Exemplo

15

- Fila de alunos:

- Matricula;

- Nome;

- Notas;

- Operações:

- Criar lista;

- Deletar fila;

- Inserir aluno;

- Remover aluno;

- Imprimir fila;

Exemplo de Fila: Definição de tipos

16

```
//Arquivo FilaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct fila Fila;

Fila* cria_Fila();
void libera_Fila(Fila* fi);
int insere_Fila(Fila* fi, struct aluno al);
int remove_Fila(Fila* fi);
void imprime_Fila(Fila* fi);
```


Exemplo de Fila: Definição de tipos

17

```
//Arquivo FilaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct fila Fila;

Fila* cria_Fila();
void libera_Fila(Fila* fi);
int insere_Fila(Fila* fi, struct aluno al);
int remove_Fila(Fila* fi);
void imprime_Fila(Fila* fi);
```

APENAS PARA OBSERVAR
E DEPURAR A FILA, NÃO É
UMA OPERAÇÃO
PADRÃO.

Exemplo de Fila: Definição de tipos

18

```
//Arquivo FilaDin.c
#include <stdio.h>
#include <stdlib.h>
#include "FilaDin.h" //inclui os Protótipos
//Definição do tipo Fila
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//Definição do Nó Descritor da Fila
struct fila{
    struct elemento *inicio;
    struct elemento *fim;
    int qtd;
};
```

Exemplo de Fila: Definição de tipos

19

```
//Arquivo FilaDin.c
#include <stdio.h>
#include <stdlib.h>
#include "FilaDin.h" //inclui os Protótipos
//Definição do tipo Fila
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//Definição do Nó Descritor da Fila
struct fila{
    struct elemento *inicio;
    struct elemento *fim;
    int qtd;
};
```

Exemplo de Fila: Criar/Liberar

20

```
Fila* cria_Fila(){
    Fila* fi = (Fila*) malloc(sizeof(Fila));
    if(fi != NULL){
        fi->fim = NULL;
        fi->inicio = NULL;
        fi->qtd = 0;
    }
    return fi;
}

void libera_Fila(Fila* fi){
    if(fi != NULL){
        Elem* no;
        while(fi->inicio != NULL){
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```

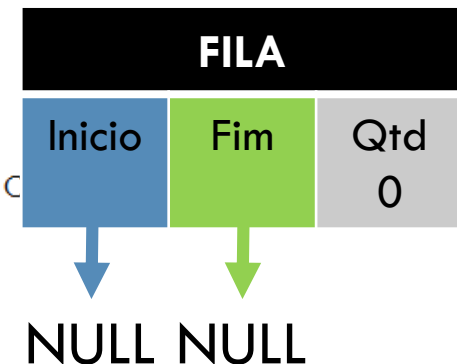
Exemplo de Fila: Criar/Liberar

21

```
Fila* cria_Fila(){
    Fila* fi = (Fila*) malloc(sizeof(Fila));
    if(fi != NULL){
        fi->fim = NULL;
        fi->inicio = NULL;
        fi->qtd = 0;
    }
    return fi;
}

void libera_Fila(Fila* fi){
    if(fi != NULL){
        Elem* no;
        while(fi->inicio != NULL){
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```

INICIALIZAR TODA A
ESTRUTURA DE
DESCRITOR.



Exemplo de Fila: Criar/Liberar

22

```
Fila* cria_Fila(){
    Fila* fi = (Fila*) malloc(sizeof(Fila));
    if(fi != NULL){
        fi->fim = NULL;
        fi->inicio = NULL;
        fi->qtd = 0;
    }
    return fi;
}

void libera_Fila(Fila* fi){
    if(fi != NULL){
        Elem* no;
        while(fi->inicio != NULL){
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```

DESALOCAR SEMPRE O
PRIMEIRO ELEMENTO, ATÉ
QUE A FILA FIQUE VAZIA.

Exemplo de Fila: Inserção

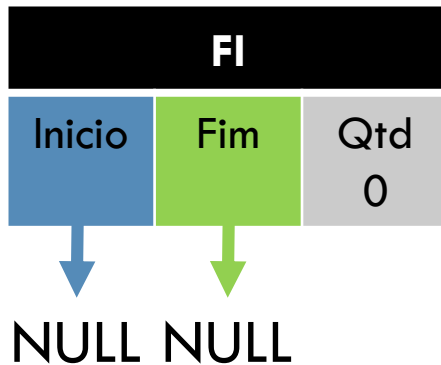
23

```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL) //fila vazia
        fi->inicio = no;
    else
        fi->fim->prox = no;
    fi->fim = no;
    fi->qtd++;
    return 1;
}
```

Exemplo de Fila: Inserção

24

AL: 32

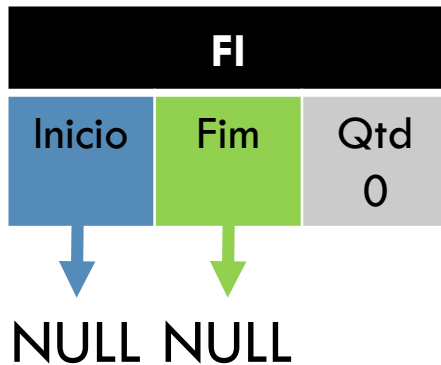


```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL) //fila vazia
        fi->inicio = no;
    else
        fi->fim->prox = no;
    fi->fim = no;
    fi->qtd++;
    return 1;
}
```


Exemplo de Fila: Inserção

25

AL: 32



```
int insere_Fila(Fila* fi, struct aluno al){  
    → if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)
```

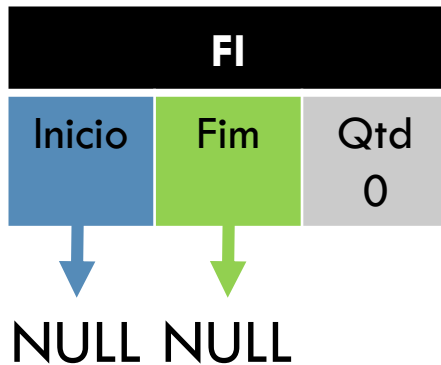
- **FILA == NULL : NÃO ALOCADA.**

```
else  
    fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

26

AL: 32

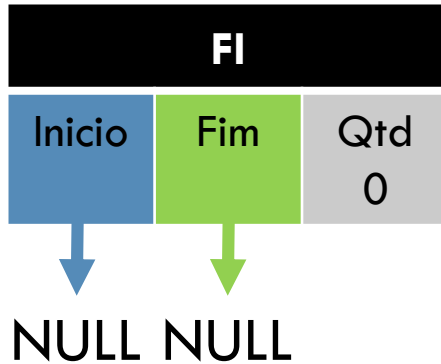


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    → Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

27

AL: 32

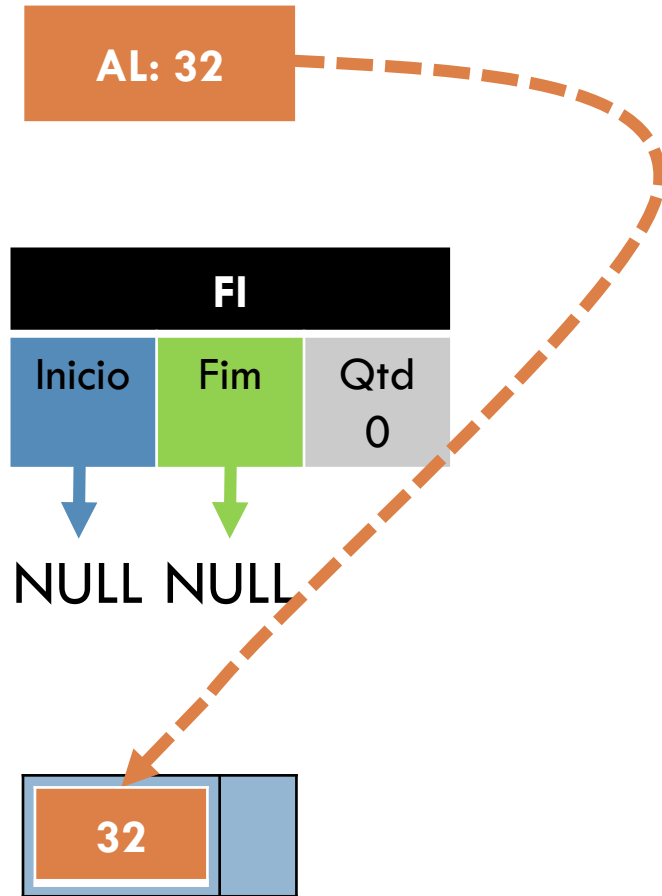


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    → if(no == NULL)  
        return 0;  
    no->dados = al;  
  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

- NO == NULL : NÃO ALOCADO.

Exemplo de Fila: Inserção

28

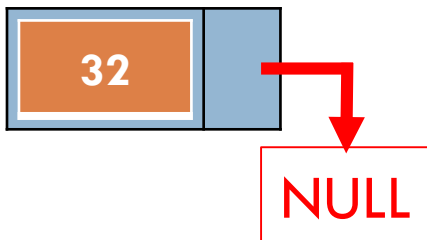
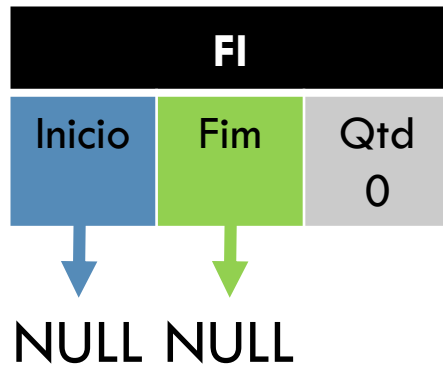


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

29

AL: 32

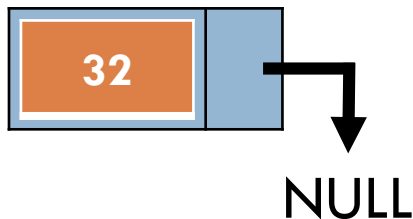
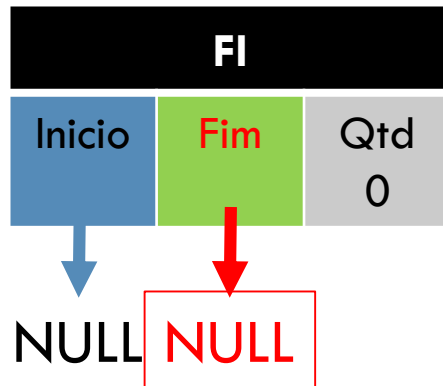


```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL) //fila vazia
        fi->inicio = no;
    else
        fi->fim->prox = no;
    fi->fim = no;
    fi->qtd++;
    return 1;
}
```

Exemplo de Fila: Inserção

30

AL: 32

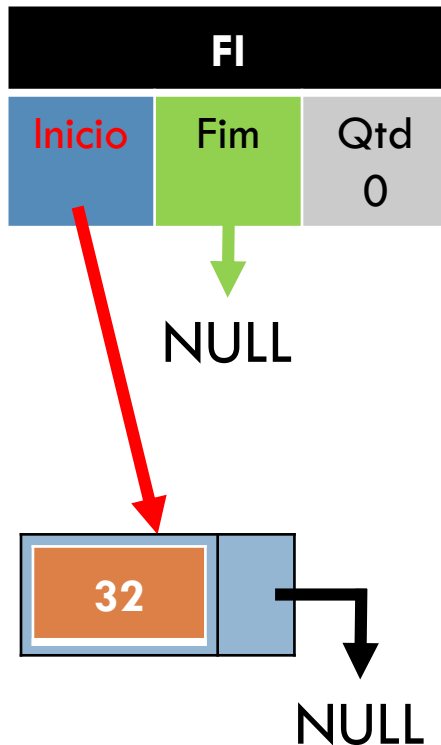


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

31

AL: 32

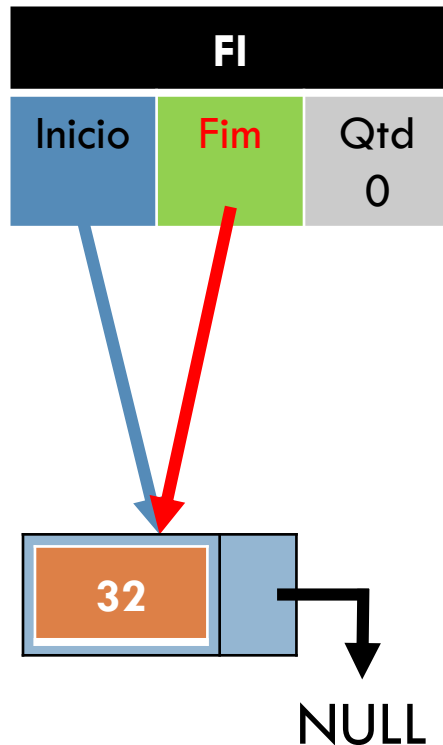


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        → fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

32

AL: 32

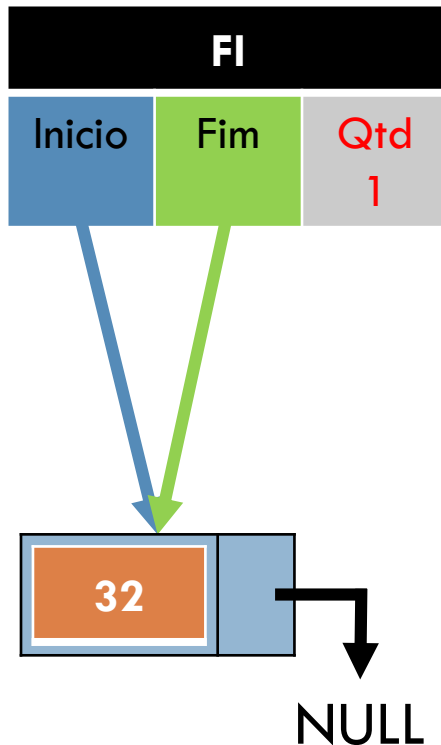


```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL) //fila vazia
        fi->inicio = no;
    else
        fi->fim->prox = no;
    → fi->fim = no;
    fi->qtd++;
    return 1;
}
```


Exemplo de Fila: Inserção

33

AL: 32

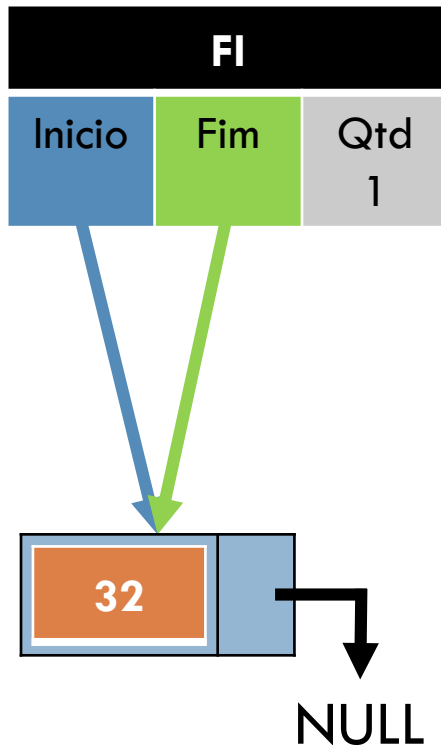


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    → fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

34

AL: 32

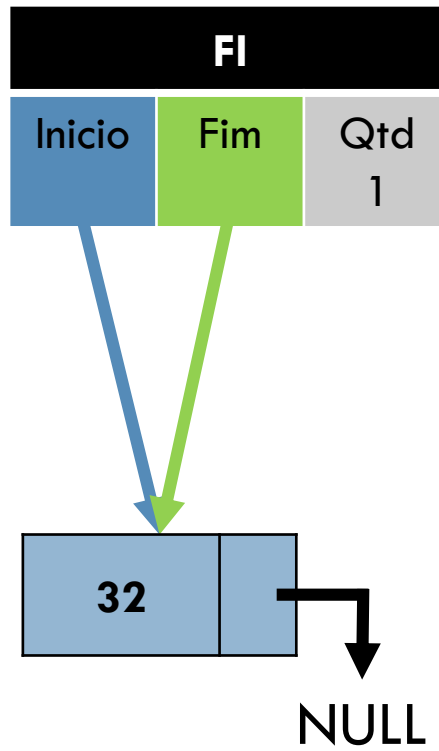


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    → return 1;  
}
```

- RETORNA OK;
- FINAL DA INSERÇÃO DO ELEMENTO.

Exemplo de Fila: Inserção

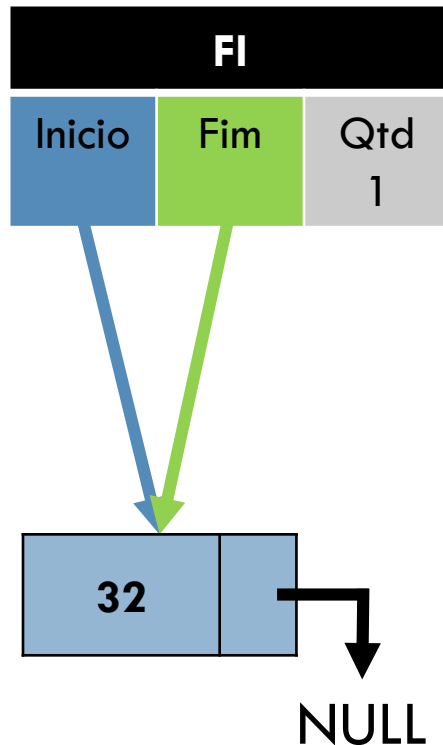
35



Exemplo de Fila: Inserção

36

AL: 98

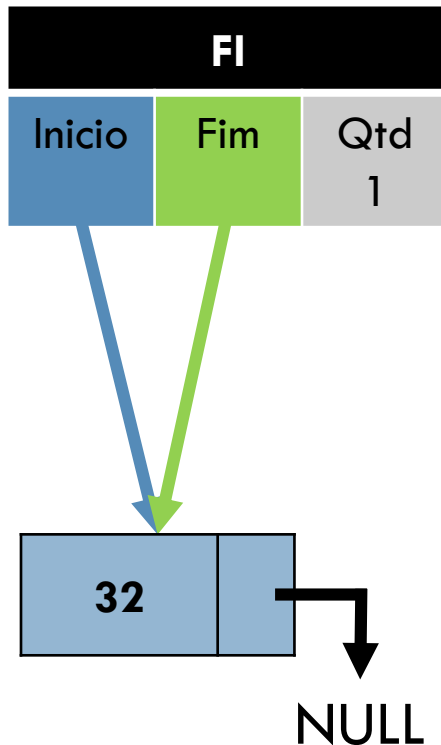


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

37

AL: 98



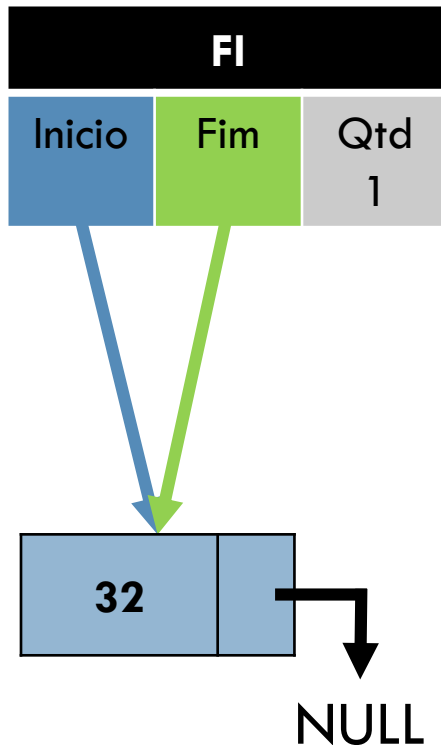
```
int insere_Fila(Fila* fi, struct aluno al){  
    → if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        else  
        fi->fim->prox = no;  
        fi->fim = no;  
        fi->qtd++;  
        return 1;  
}
```

- **FILA == NULL : NÃO ALOCADA.**

Exemplo de Fila: Inserção

38

AL: 98

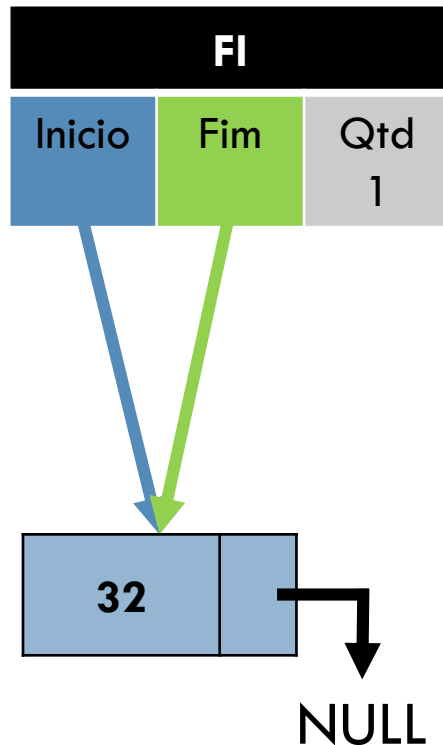


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

39

AL: 98

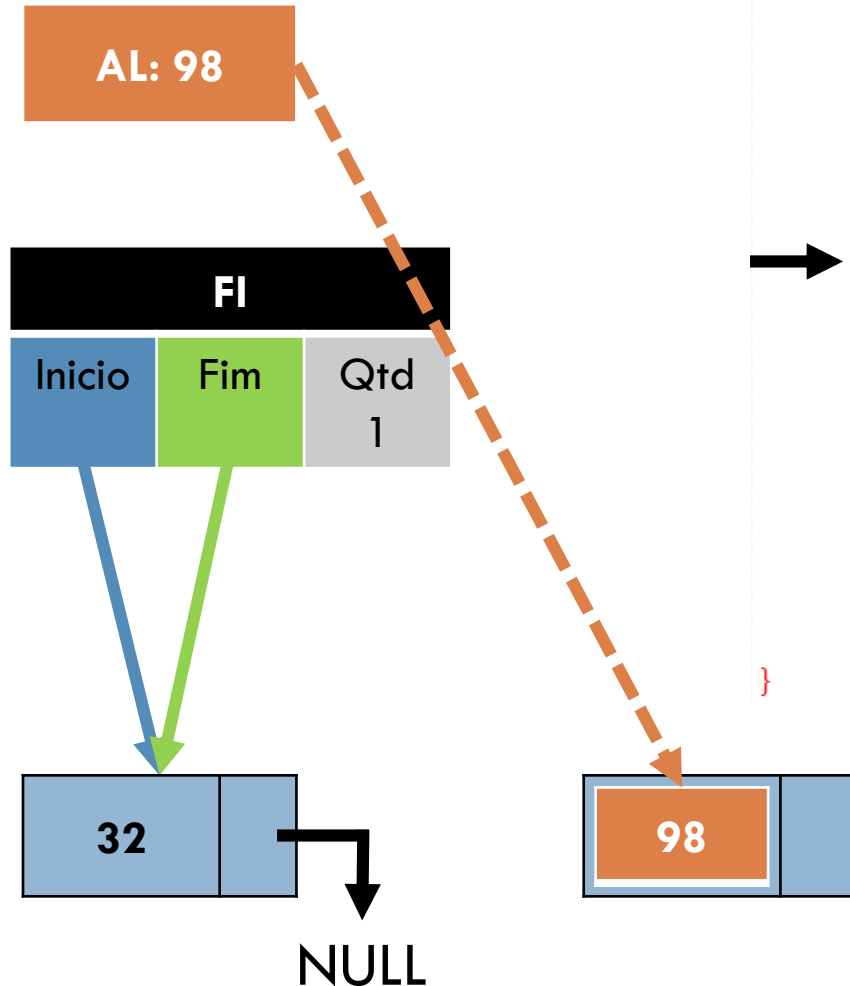


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    → if(no == NULL)  
        return 0;  
    no->dados = al;  
  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

- NO == NULL : NÃO ALOCADO.

Exemplo de Fila: Inserção

40

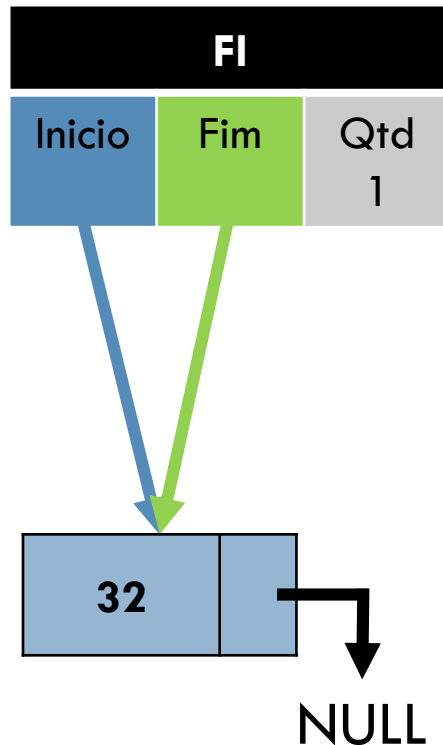


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

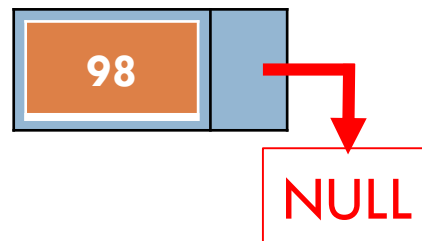

Exemplo de Fila: Inserção

41

AL: 98



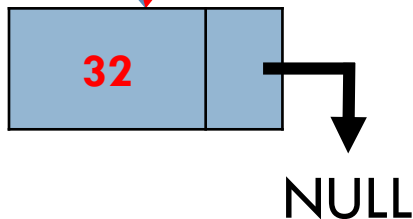
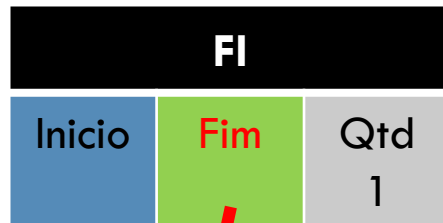
```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```



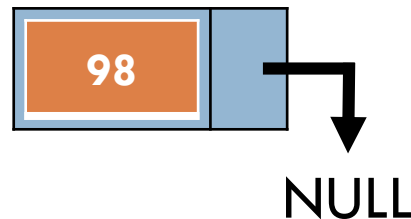
Exemplo de Fila: Inserção

42

AL: 98



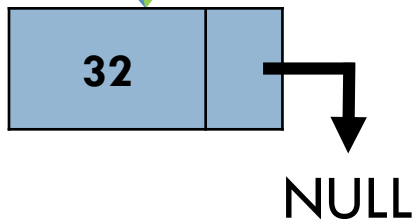
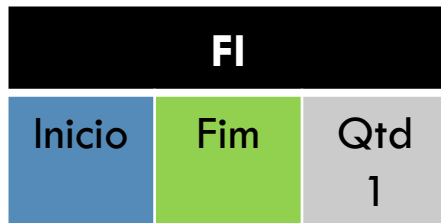
```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    → if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```



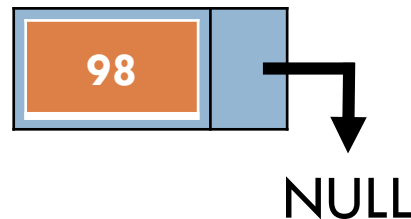
Exemplo de Fila: Inserção

43

AL: 98



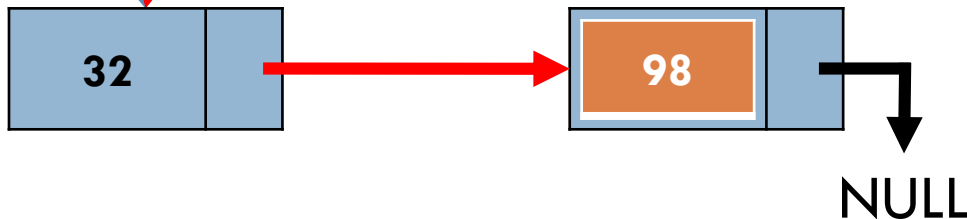
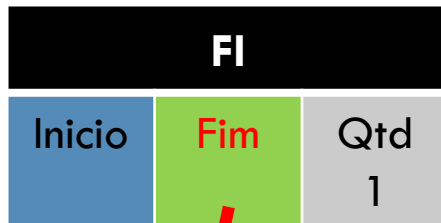
```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```



Exemplo de Fila: Inserção

44

AL: 98

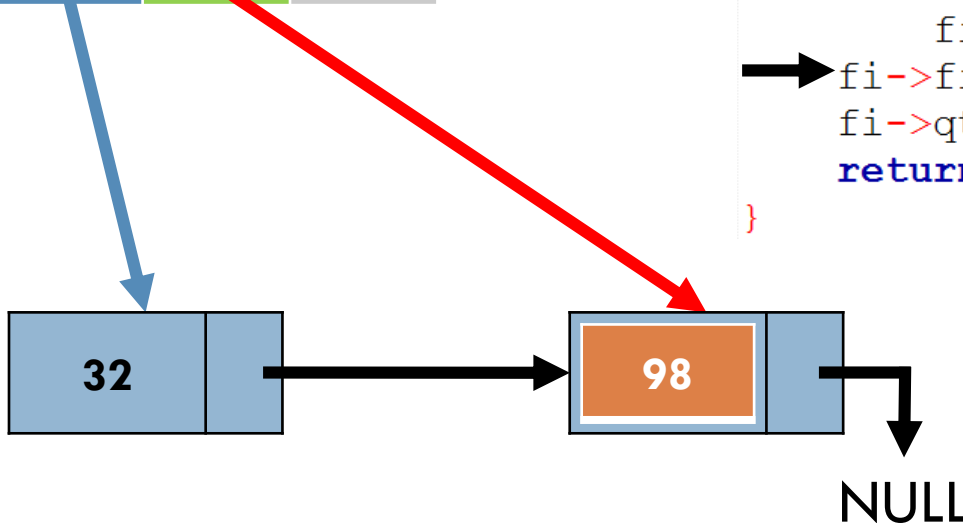


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

45

AL: 98

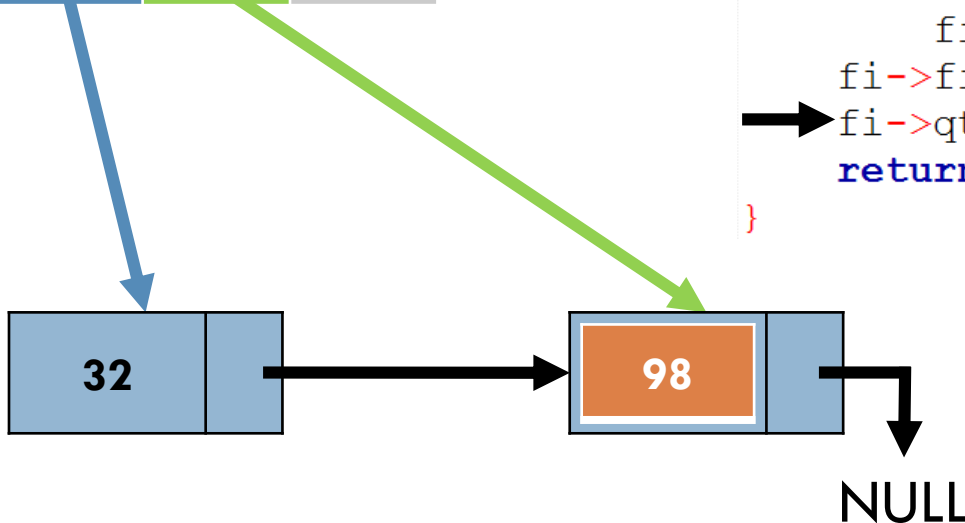
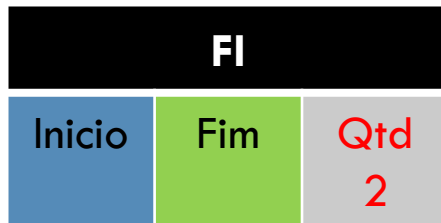


```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

46

AL: 98



```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = NULL;  
    if(fi->fim == NULL) //fila vazia  
        fi->inicio = no;  
    else  
        fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    return 1;  
}
```

Exemplo de Fila: Inserção

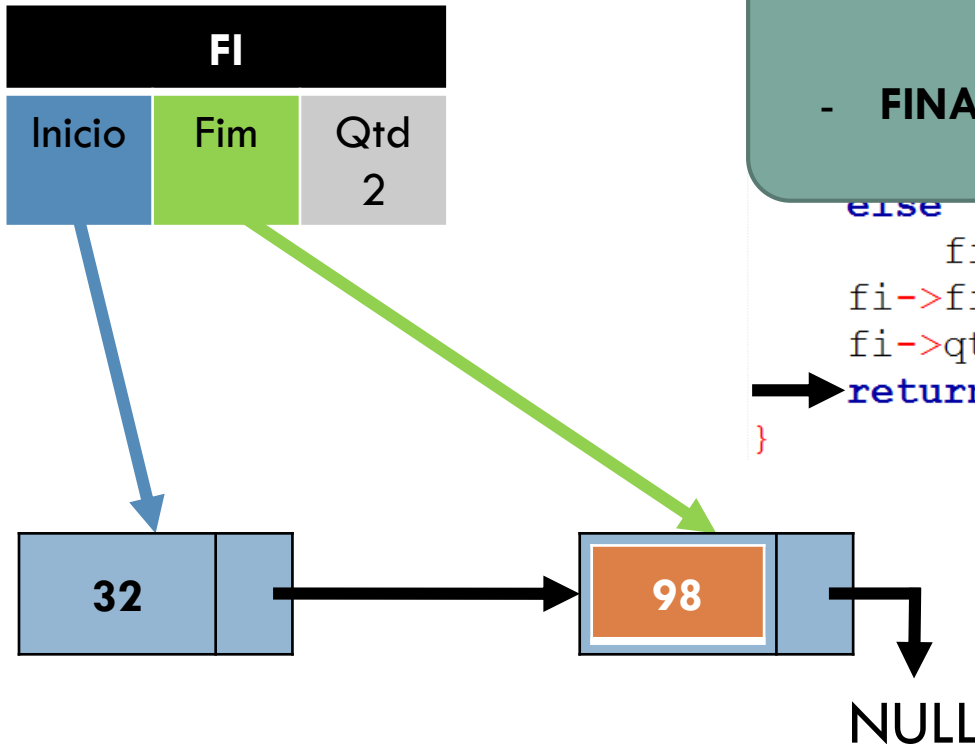
47

AL: 98

```
int insere_Fila(Fila* fi, struct aluno al){  
    if(fi == NULL)  
        return 0;  
    Elem *no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)
```

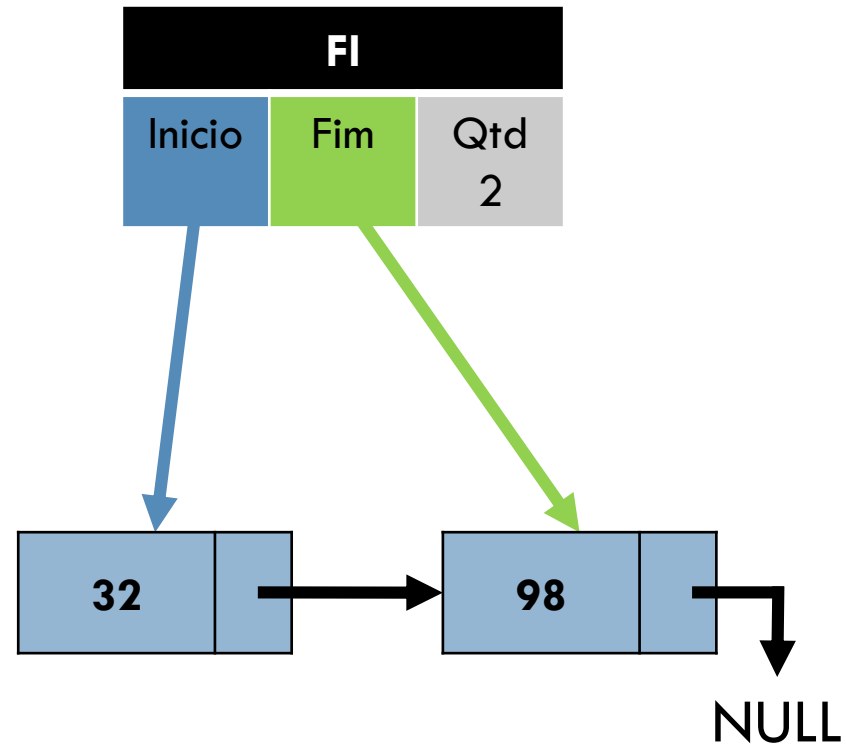
- RETORNA OK;
- FINAL DA INSERÇÃO DO ELEMENTO.

```
else  
    fi->fim->prox = no;  
    fi->fim = no;  
    fi->qtd++;  
    → return 1;  
}
```



Exemplo de Fila: Inserção

48



Exemplo de Fila: Remoção

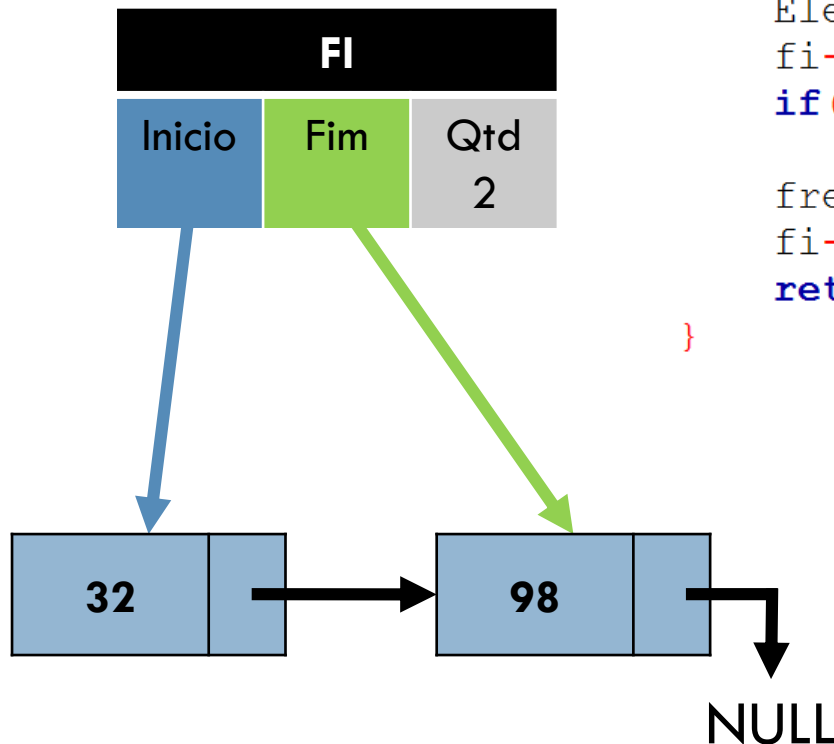
49

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL) //fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL) //fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}
```

Exemplo de Fila: Remoção

50

```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```

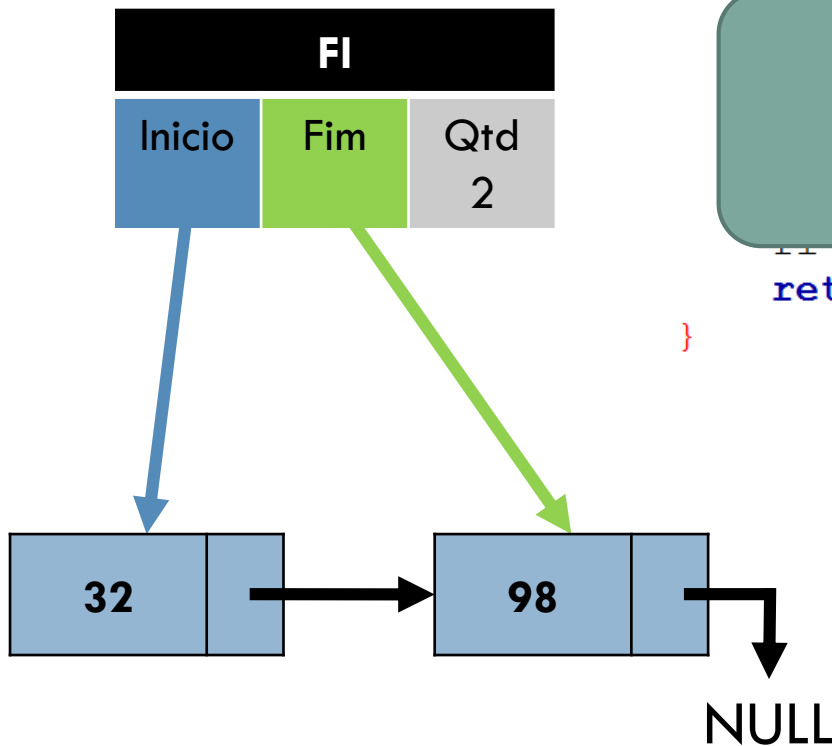


Exemplo de Fila: Remoção

51

```
int remove_Fila(Fila* fi){  
    → if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;
```

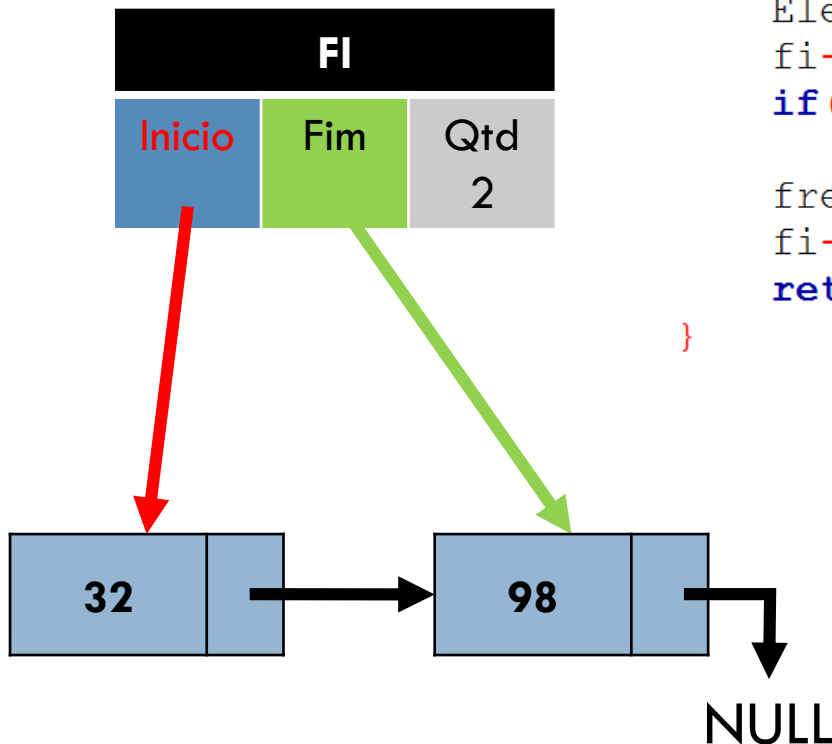
- **FILA == NULL : NÃO ALOCADA.**



Exemplo de Fila: Remoção

52

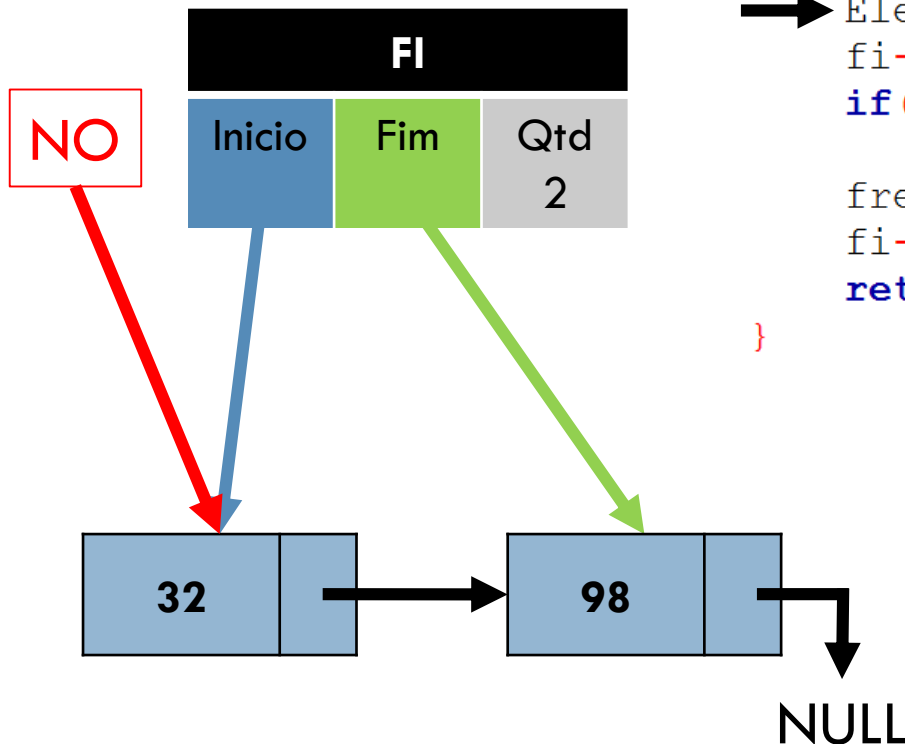
```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    → if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```



Exemplo de Fila: Remoção

53

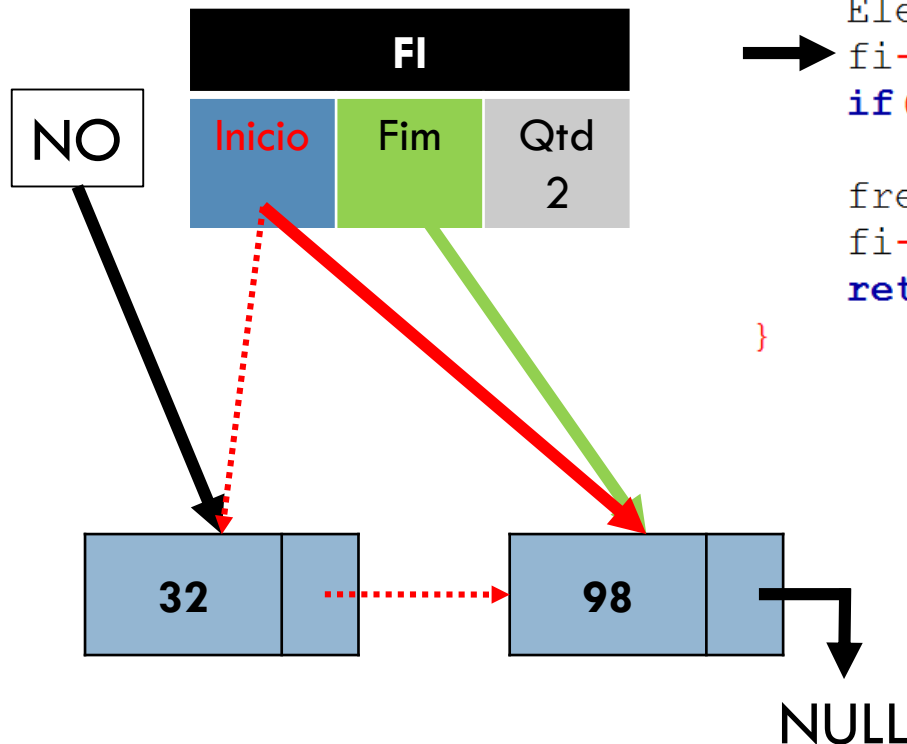
```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```



Exemplo de Fila: Remoção

54

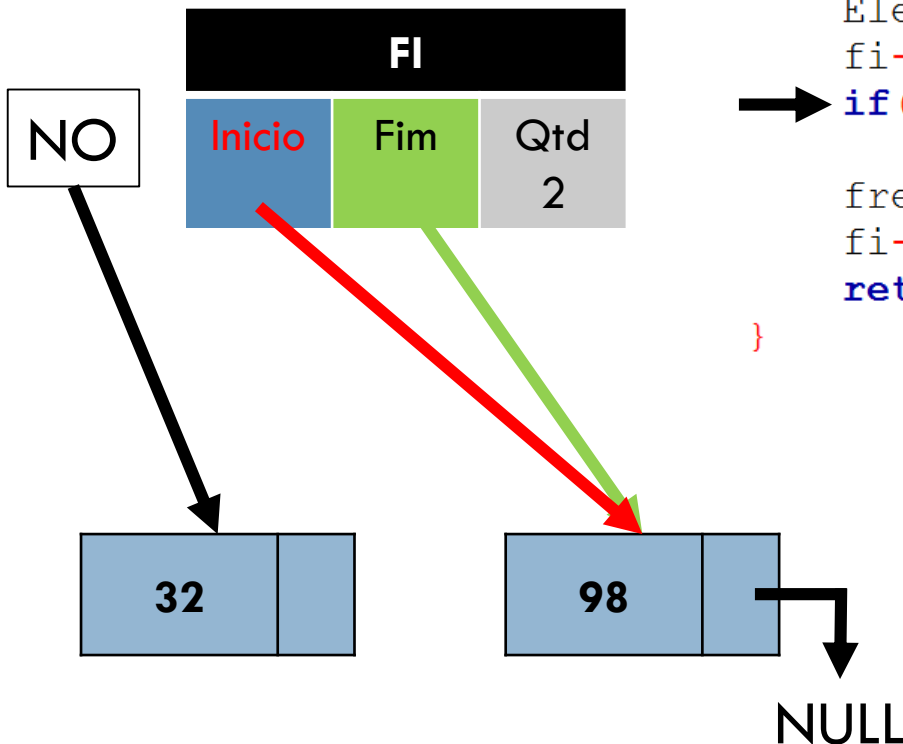
```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```



Exemplo de Fila: Remoção

55

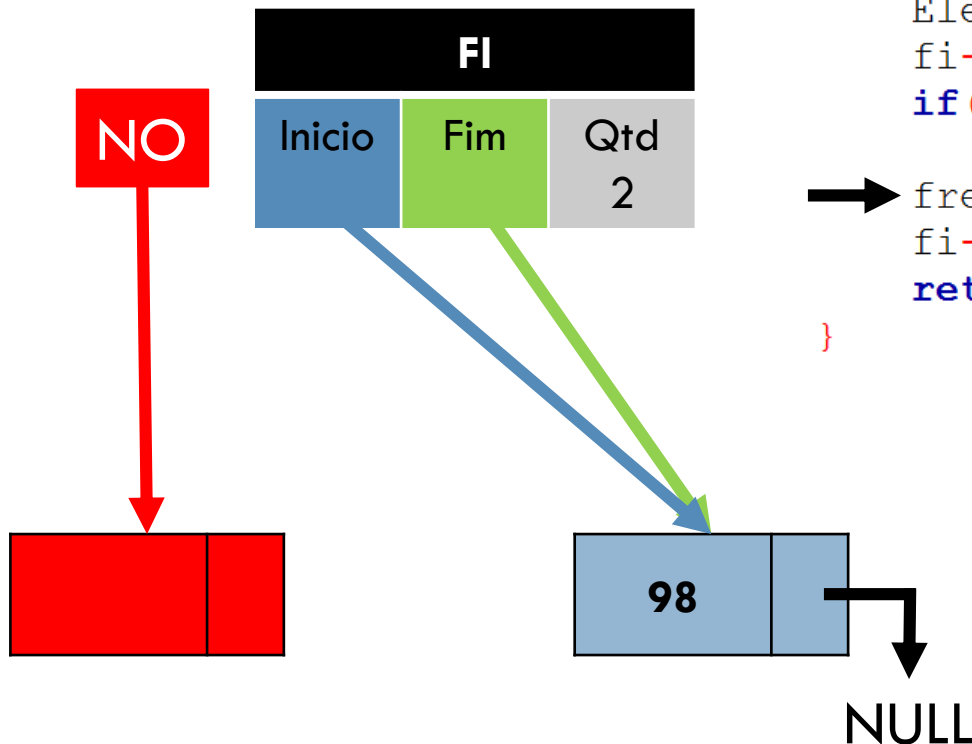
```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    → if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```



Exemplo de Fila: Remoção

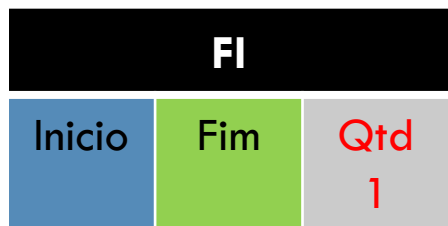
56

```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```



Exemplo de Fila: Remoção

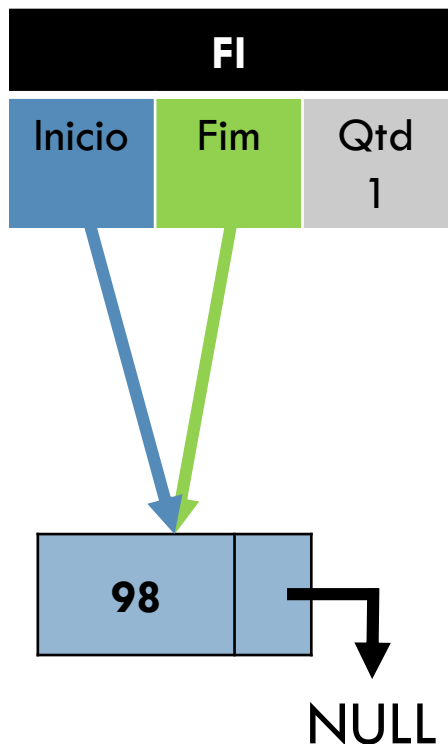
57



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

58

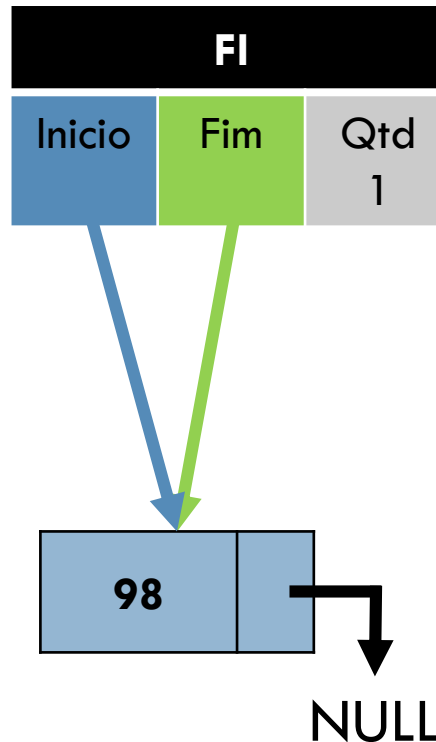


```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    → return 1;  
}
```

- RETORNA OK;
- FINAL DA REMOÇÃO DO ELEMENTO.

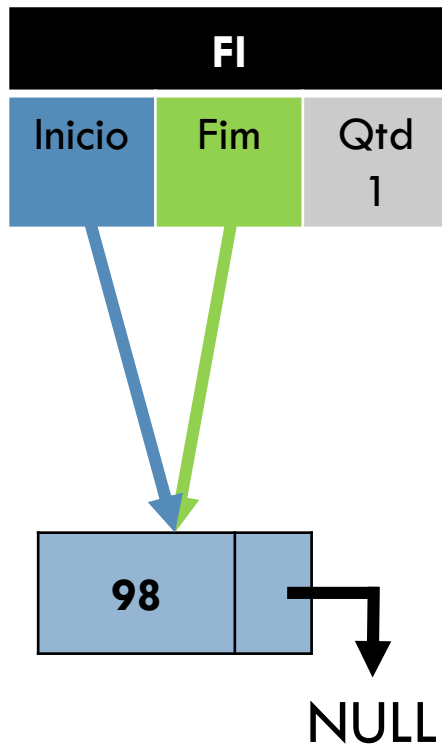
Exemplo de Fila: Remoção

59



Exemplo de Fila: Remoção

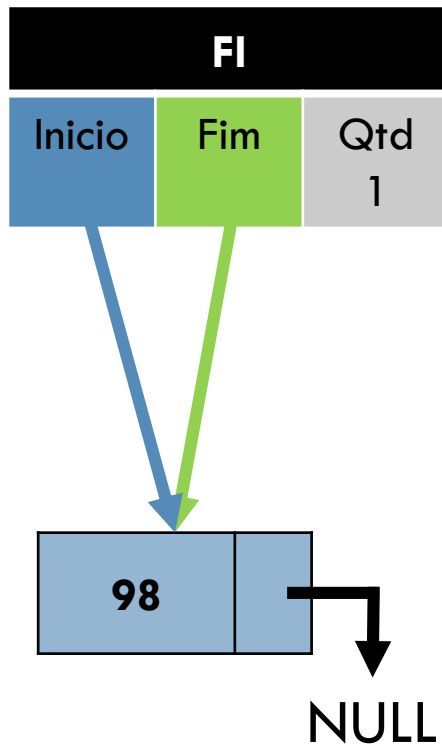
60



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

61



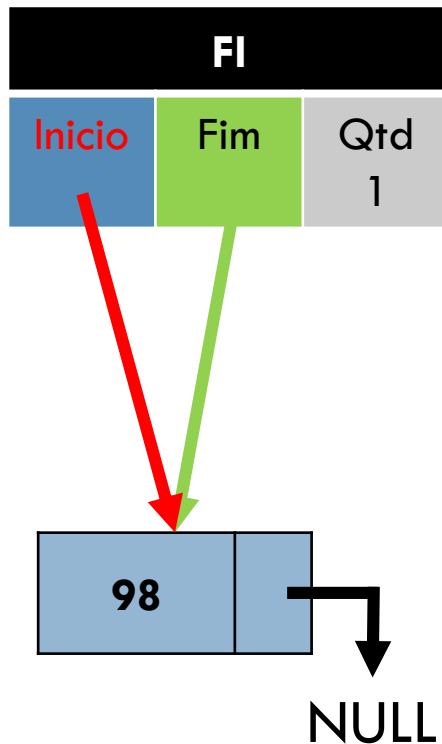
```
int remove_Fila(Fila* fi){  
    → if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;
```

- **FILA == NULL : NÃO ALOCADA.**

```
    if(fi->qtd > 0)  
        return 1;  
}
```

Exemplo de Fila: Remoção

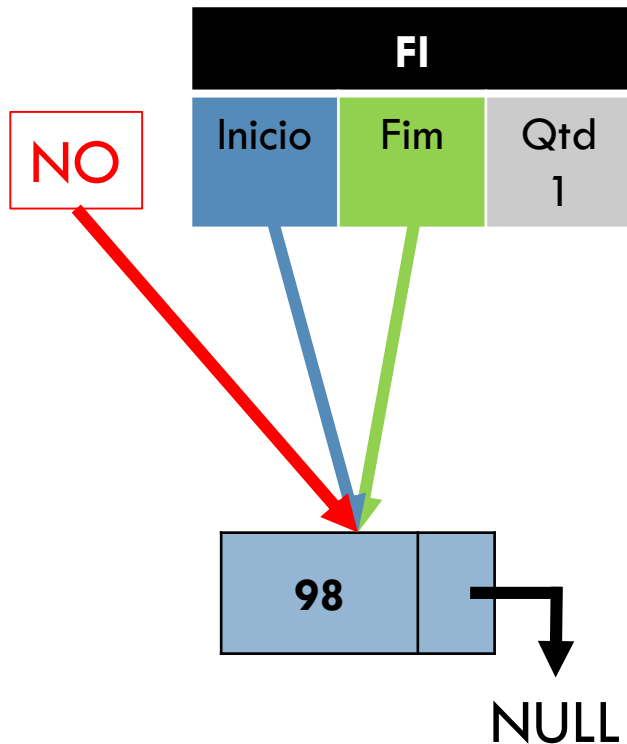
62



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    → if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

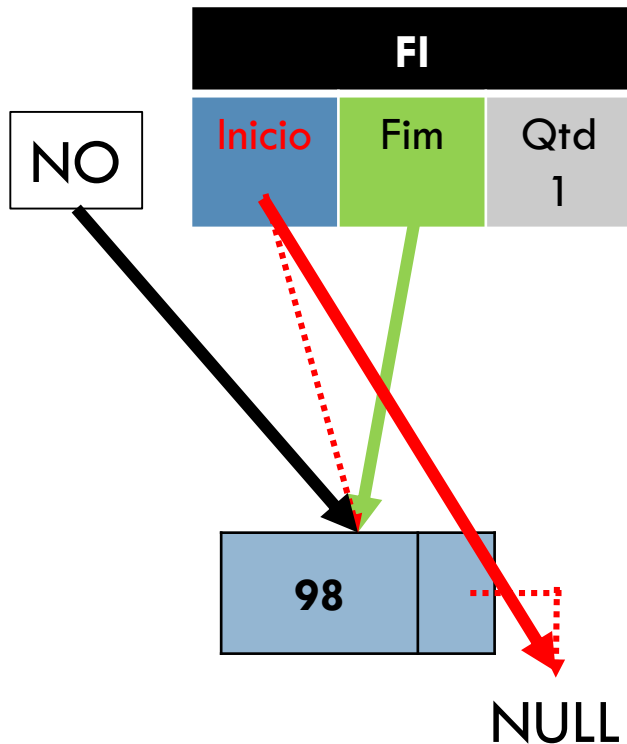
63



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    → Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

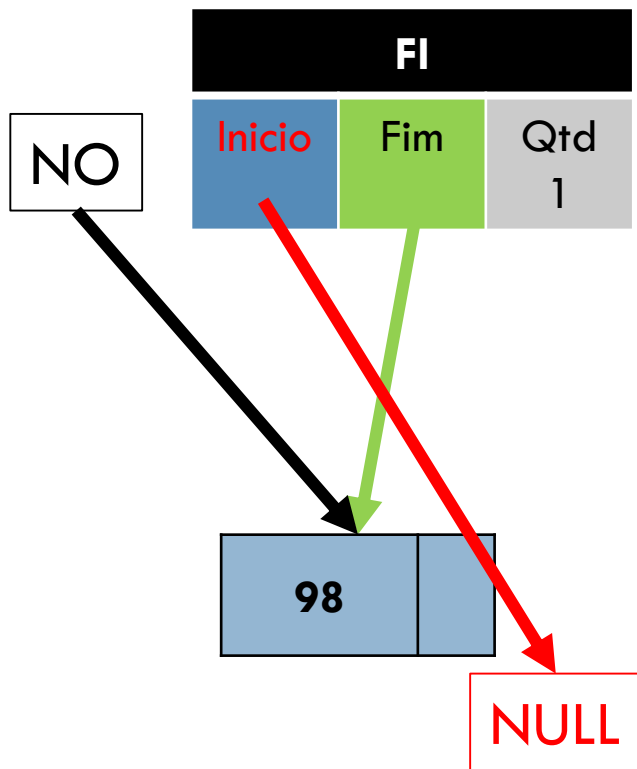
64



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    → fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```


Exemplo de Fila: Remoção

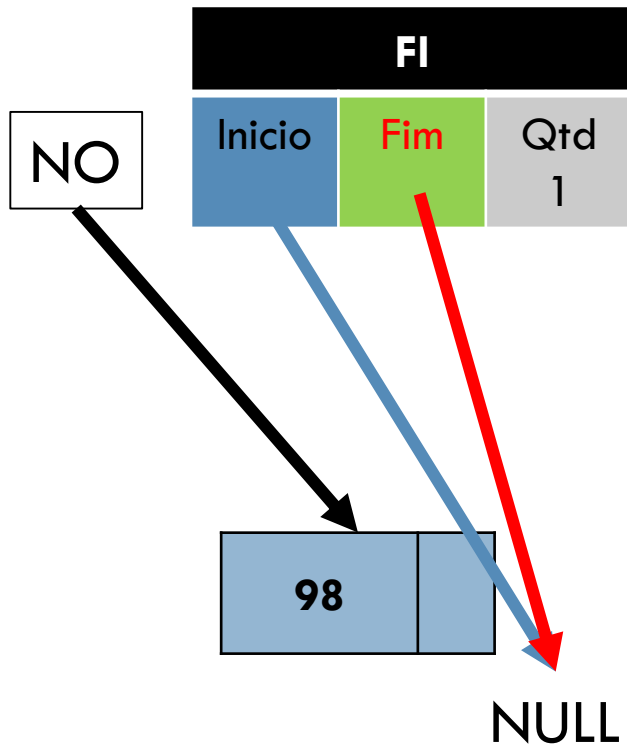
65



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    → if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

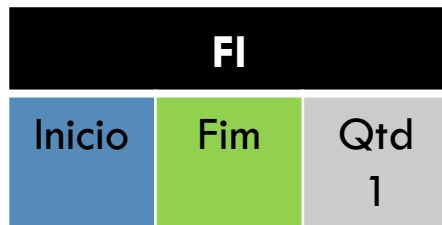
66



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        → fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

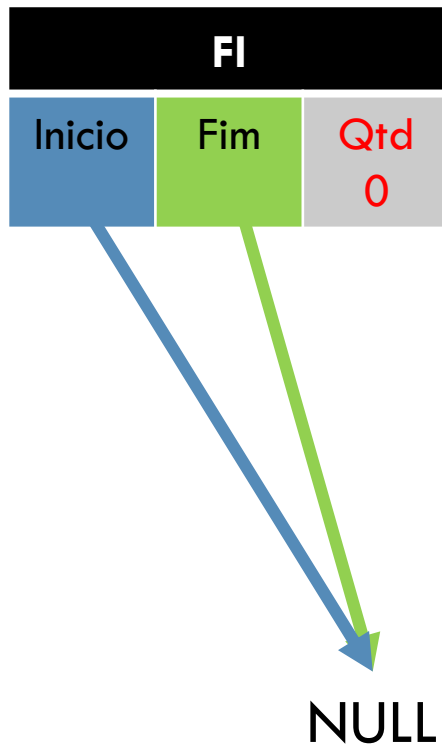
67



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    → free(no);  
    fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

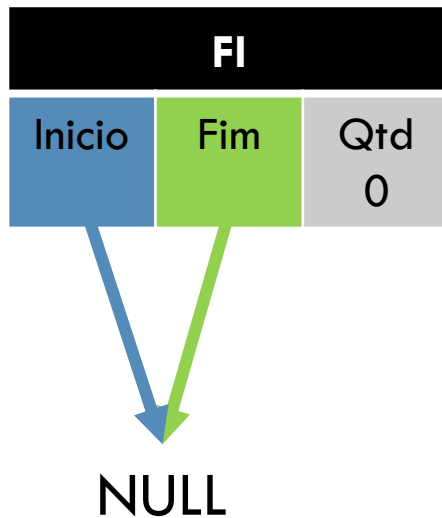
68



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    → fi->qtd--;  
    return 1;  
}
```

Exemplo de Fila: Remoção

69



```
int remove_Fila(Fila* fi){  
    if(fi == NULL)  
        return 0;  
    if(fi->inicio == NULL)//fila vazia  
        return 0;  
    Elem *no = fi->inicio;  
    fi->inicio = fi->inicio->prox;  
    if(fi->inicio == NULL)//fila ficou vazia  
        fi->fim = NULL;  
    free(no);  
    fi->qtd--;  
    → return 1;  
}
```

- RETORNA OK;
- FINAL DA REMOÇÃO DO ELEMENTO.

Fila Estática

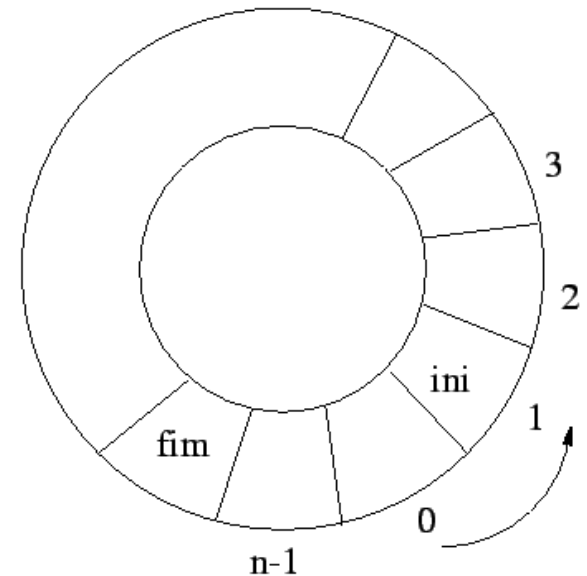
70

- ❑ Armazena em vetores.
- ❑ Utiliza índice para conhecer o início e o final da fila.
- ❑ Toda remoção gera um deslocamento dos elementos.

Fila Estática

71

- Armazena em vetores.
- Utiliza índice para conhecer o início e o final da fila.
- Toda remoção gera um deslocamento dos elementos.
 - ▣ Solução: Fila circular estática.
 - ▣ Variação da Lista circular.



Fila: Exercícios

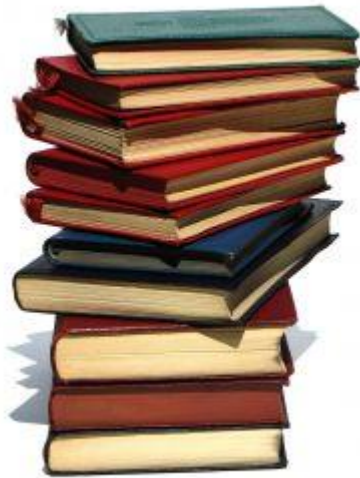
72

- Crie uma função de consulta para a Fila do exemplo. **A consulta só é realizada no elemento inicial.**
- Adapte a lista estática para funcionar como uma fila.
- Como realizar as operações de inserção e remoção em uma **fila circular estática**?
 - ▣ Quem se torna o primeiro na remoção?
 - ▣ Quem se torna o final na inserção?

Pilha

73

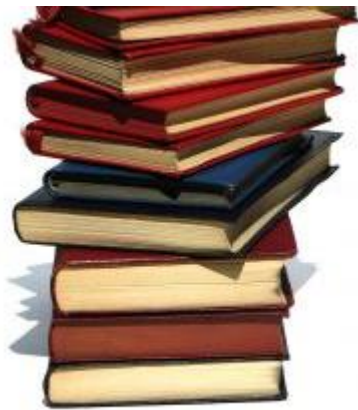
- Tipo especial de lista.
- Ordem de inserção e remoção.



Pilha

74

- Tipo especial de lista.
- Ordem de inserção e remoção.



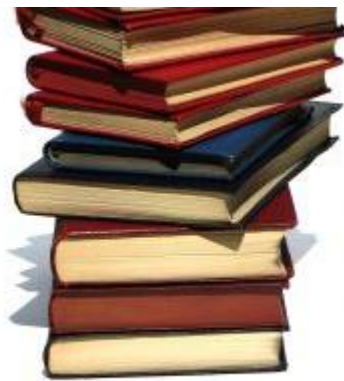
Remoção



Pilha

75

- Tipo especial de lista.
- Ordem de inserção e remoção.



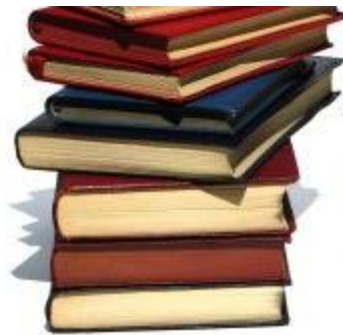
Remoção



Pilha

76

- Tipo especial de lista.
- Ordem de inserção e remoção.



Remoção

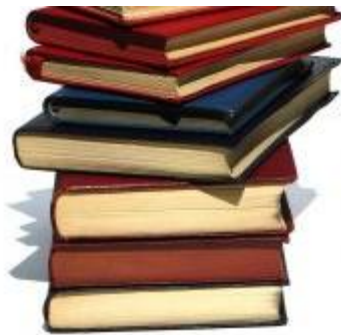


Pilha

77

- Tipo especial de lista.
- Ordem de inserção e remoção.

Inserção



Pilha

78

- Tipo especial de lista.
- Ordem de inserção e remoção.

Inserção



Pilha

79

- Tipo especial de lista.
- Ordem de inserção e remoção.

Inserção



Pilha

80

- Tipo especial de lista.
- Ordem de inserção e remoção.

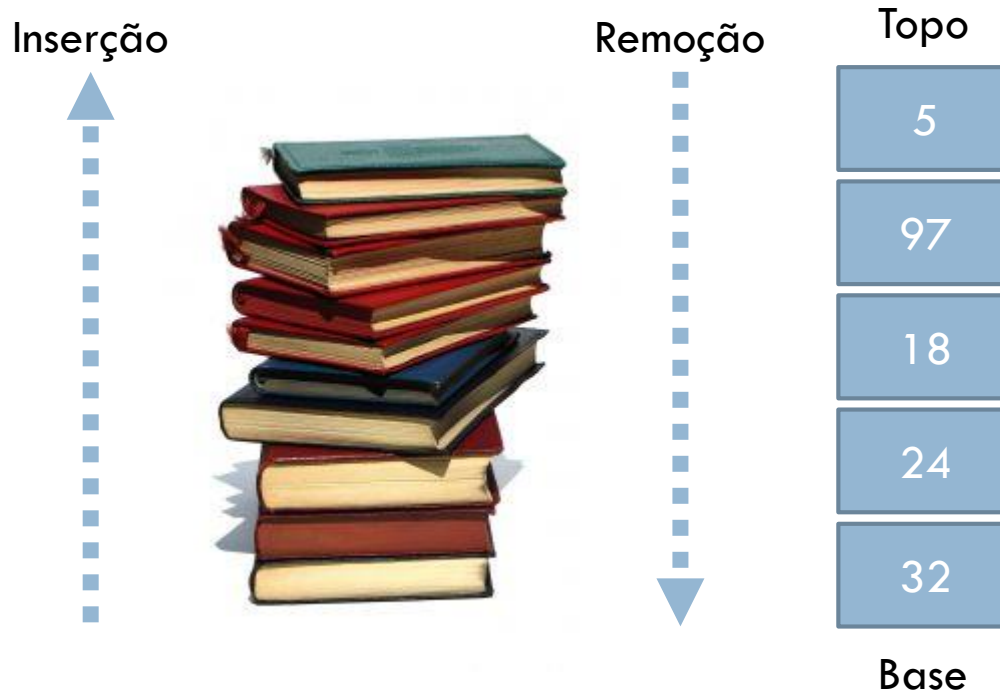
Inserção



Pilha

81

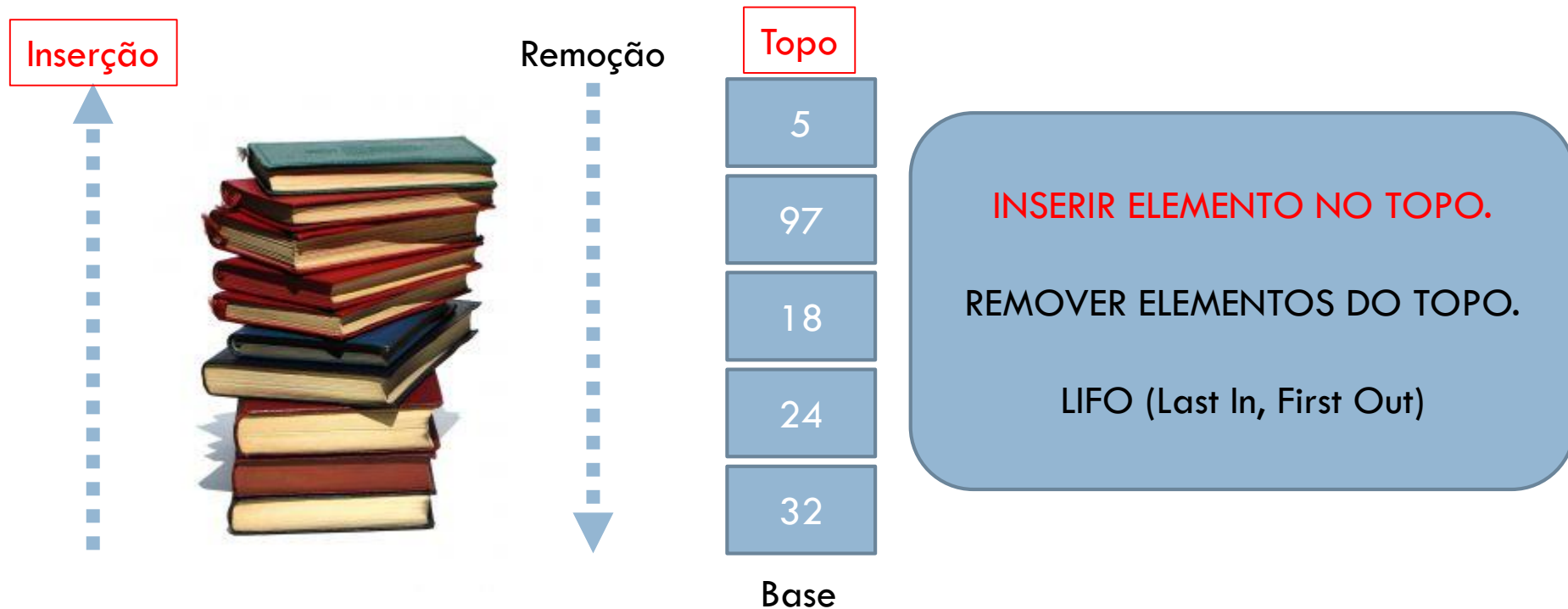
- Tipo especial de lista.
- Ordem de inserção e remoção.



Pilha

82

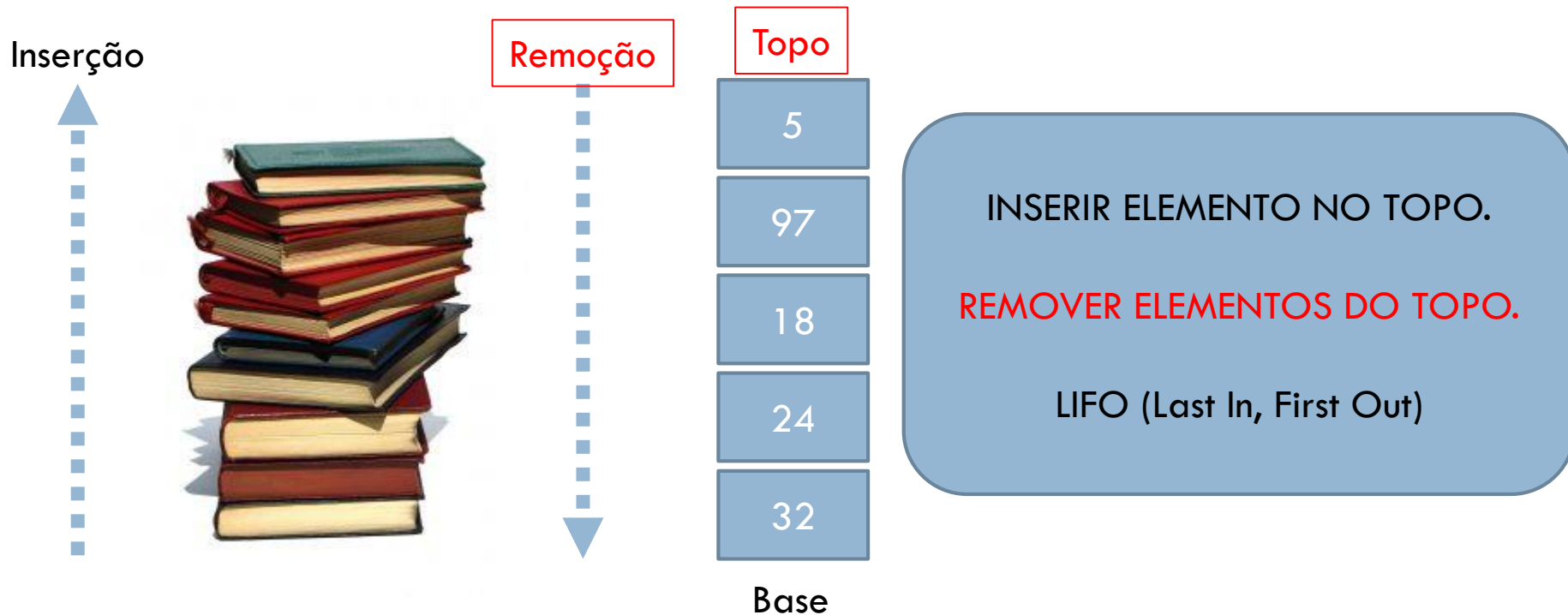
- Tipo especial de lista.
- Ordem de inserção e remoção.



Pilha

83

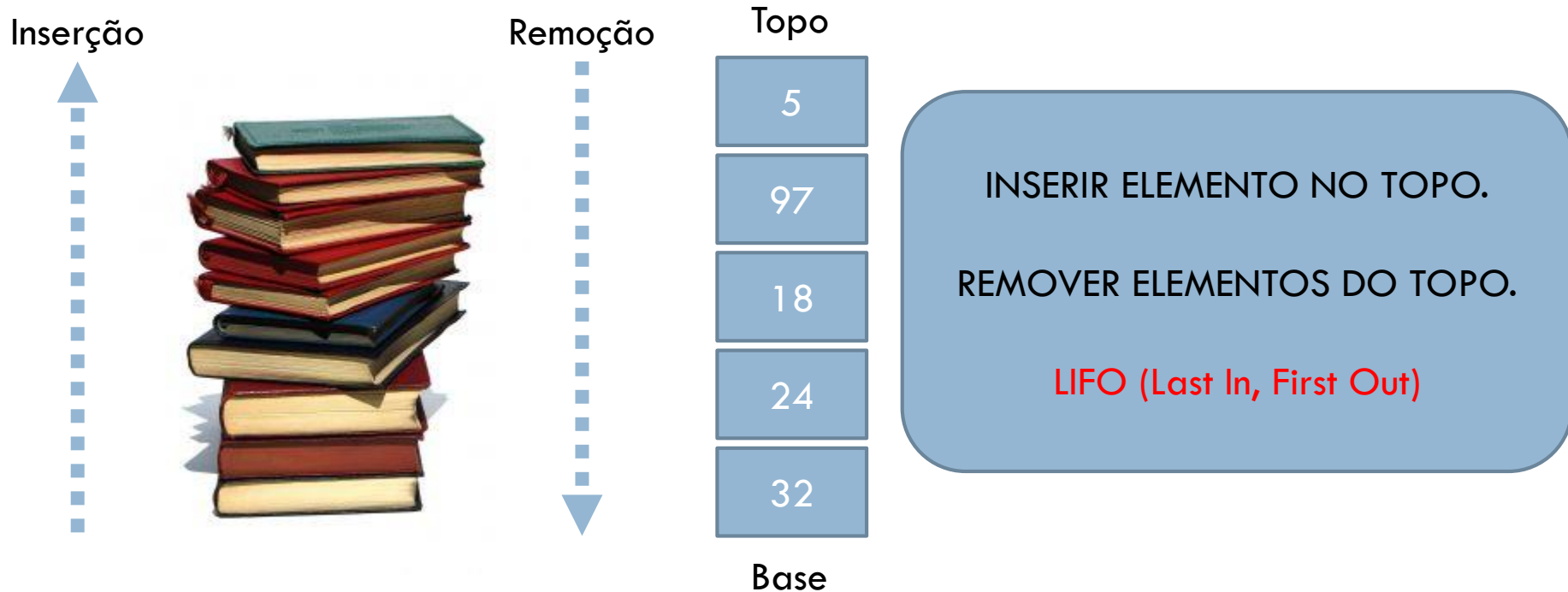
- Tipo especial de lista.
- Ordem de inserção e remoção.



Pilha

84

- Tipo especial de lista.
- Ordem de inserção e remoção.



Pilha

85

- Aplicações:
 - ▣ Análise de expressões matemáticas;
 - ▣ Conversão de base de valores.

Pilha

86

- Operações comuns de uma Pilha:
 - ▣ Criação da pilha (Alocar);
 - ▣ Inserção de um elemento no início (EMPILHAR);
 - ▣ Remoção de um elemento do início (DESEMPILHAR);
 - ▣ Acesso ao elemento (Início);
 - ▣ Destruição da fila (Desalocar).

Pilha

87

- Operações comuns de uma Pilha:
 - ▣ Criação da pilha (Alocar);
 - ▣ Inserção de um elemento no início (EMPILHAR);
 - ▣ Remoção de um elemento do início (DESEMPILHAR);
 - ▣ Acesso ao elemento (Início);
 - ▣ Destruição da fila (Desalocar).

- Pilha estática: derivada de listas estáticas.
- Pilha dinâmica: derivada de listas encadeadas simples.

Pilha Dinâmica: Exemplo

88

- Fila de alunos:
 - Matricula;
 - Nome;
 - Notas;
- Operações:
 - Criar lista;
 - Deletar fila;
 - Inserir aluno;
 - Remover aluno;
 - Imprimir fila;

Exemplo de Pilha: Definição de tipos

89

```
//Arquivo PilhaDin.h
```

```
struct aluno{  
    int matricula;  
    char nome[30];  
    float n1,n2,n3;  
};
```

```
typedef struct elemento* Pilha;
```

```
Pilha* cria_Pilha();
```

```
void libera_Pilha(Pilha* pi);
```

```
int insere_Pilha(Pilha* pi, struct aluno al);
```

```
int remove_Pilha(Pilha* pi);
```

```
void imprime_Pilha(Pilha* pi);
```

Exemplo de Pilha: Definição de tipos

90

```
//Arquivo PilhaDin.h
```

```
struct aluno{  
    int matricula;  
    char nome[30];  
    float n1,n2,n3;  
};
```

IGUAL LISTA ENCADEADA DINÂMICA.

```
typedef struct elemento* Pilha;
```

```
Pilha* cria_Pilha();
```

```
void libera_Pilha(Pilha* pi);
```

```
int insere_Pilha(Pilha* pi, struct aluno al);
```

```
int remove_Pilha(Pilha* pi);
```

```
void imprime_Pilha(Pilha* pi);
```

Exemplo de Pilha: Definição de tipos

91

```
//Arquivo PilhaDin.c

#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h" //inclui os Protótipos

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
```

Exemplo de Pilha: Definição de tipos

92

```
//Arquivo PilhaDin.c
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include "PilhaDin.h" //inclui os Protótipos
```

```
//Definição do tipo Pilha
```

```
struct elemento{  
    struct aluno dados;  
    struct elemento *prox;  
};
```

```
typedef struct elemento Elem;
```

**IGUAL LISTA ENCADEADA
DINÂMICA.**

Exemplo de Pilha: Criar/Liberar

93

```
Pilha* cria_Pilha() {
    Pilha* pi = (Pilha*) malloc(sizeof(Pilha));
    if(pi != NULL)
        *pi = NULL;
    return pi;
}

void libera_Pilha(Pilha* pi) {
    if(pi != NULL) {
        Elem* no;
        while((*pi) != NULL) {
            no = *pi;
            *pi = (*pi)->prox;
            free(no);
        }
        free(pi);
    }
}
```

Exemplo de Pilha: Criar/Liberar

94

```
Pilha* cria_Pilha() {
    Pilha* pi = (Pilha*) malloc(sizeof(Pilha));
    if(pi != NULL)
        *pi = NULL;
    return pi;
}

void libera_Pilha(Pilha* pi) {
    if(pi != NULL) {
        Elem* no;
        while((*pi) != NULL) {
            no = *pi;
            *pi = (*pi)->prox;
            free(no);
        }
        free(pi);
    }
}
```

**IGUAL LISTA ENCADEADA
DINÂMICA.**

Exemplo de Pilha: Inserção

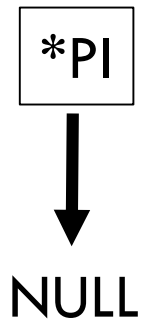
95

```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = (*pi);
    *pi = no;
    return 1;
}
```

Exemplo de Pilha: Inserção

96

AL: 32



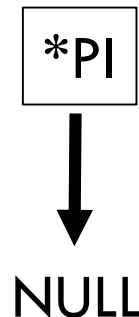
```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = (*pi);  
    *pi = no;  
    return 1;  
}
```


Exemplo de Pilha: Inserção

97

AL: 32

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    → if(pi == NULL)  
        return 0;  
    Elem* no;  
  
    PILHA == NULL : NÃO ALOCADA.  
  
    *pi = no;  
    return 1;  
}
```

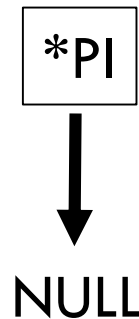


Exemplo de Pilha: Inserção

98

AL: 32

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    → { Elem* no;  
        no = (Elem*) malloc(sizeof(Elem));  
        if(no == NULL)  
            return 0;  
        no->dados = al;  
        no->prox = (*pi);  
        *pi = no;  
        return 1;  
    }
```

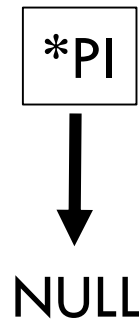


Exemplo de Pilha: Inserção

99

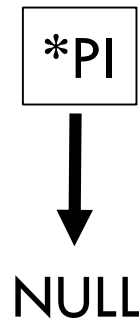
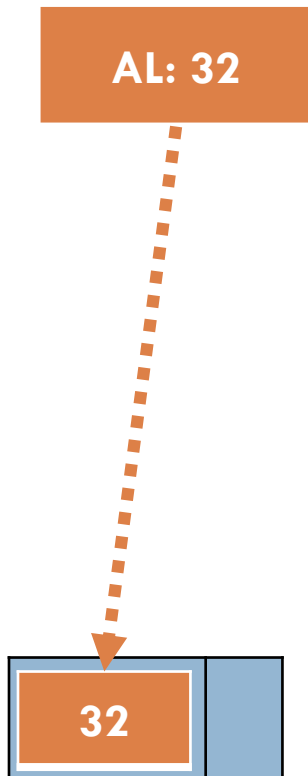
AL: 32

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    → if(no == NULL)  
        NO == NULL : NÃO ALOCADO.  
}
```



Exemplo de Pilha: Inserção

100

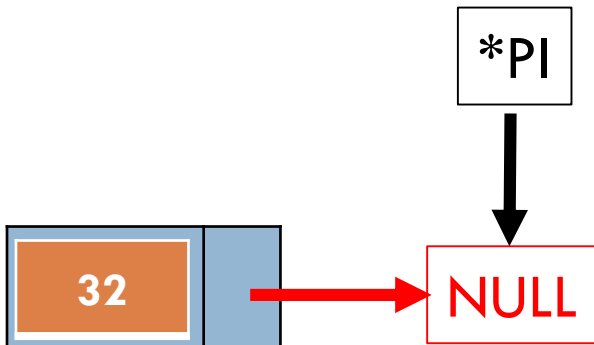


```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    → no->dados = al;  
    no->prox = (*pi);  
    *pi = no;  
    return 1;  
}
```

Exemplo de Pilha: Inserção

101

AL: 32



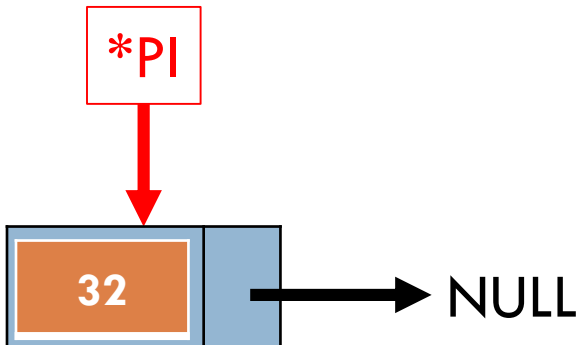
```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    → no->prox = (*pi);  
    *pi = no;  
    return 1;  
}
```

SEMPRE APONTA PARA O INICIO DA PILHA.

Exemplo de Pilha: Inserção

102

AL: 32

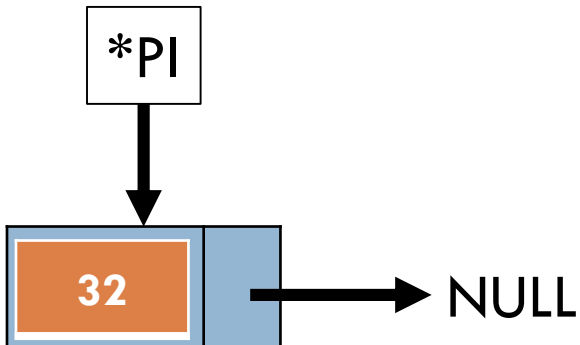


```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = (*pi);  
    → *pi = no;  
    return 1;  
}
```

Exemplo de Pilha: Inserção

103

AL: 32



```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = (*pi);  
    *pi = no;  
    → return 1;  
}
```

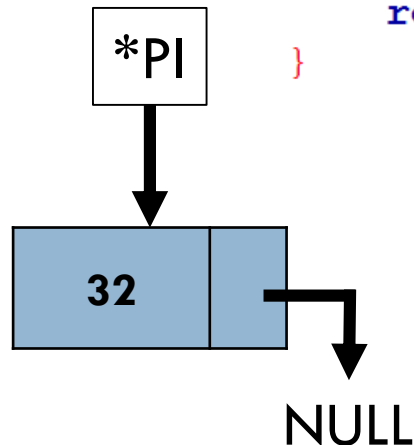
- RETORNA OK;
- FINAL DA INSERÇÃO DO ELEMENTO.

Exemplo de Pilha: Inserção

104

AL: 98

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = (*pi);  
    *pi = no;  
    return 1;  
}
```



Exemplo de Pilha: Inserção

105

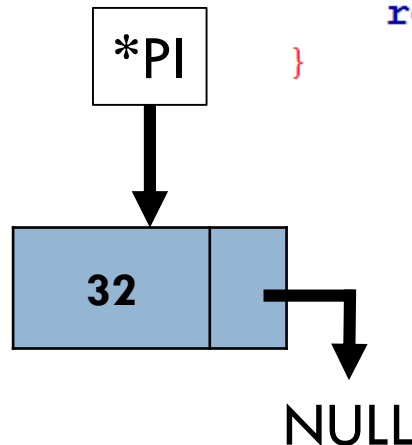
AL: 98

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    → if(pi == NULL)  
        return 0;  
    Elem* no;
```

PILHA == NULL : NÃO ALOCADA.

```
    *pi = no;  
    return 1;
```

```
}
```

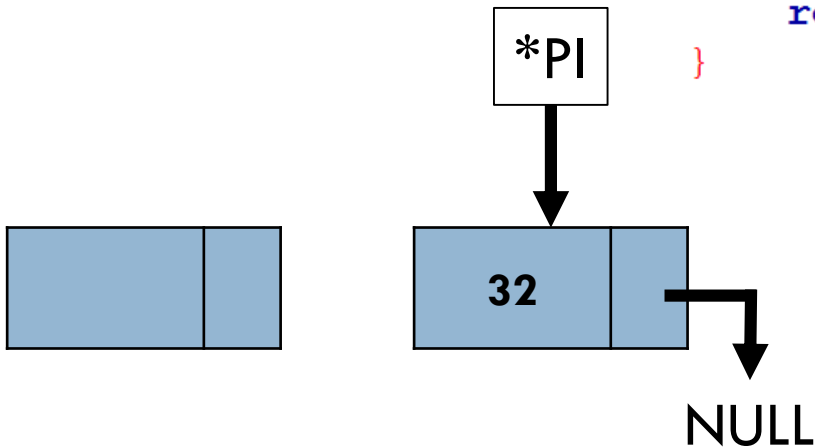


Exemplo de Pilha: Inserção

106

AL: 98

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    → { Elem* no;  
        no = (Elem*) malloc(sizeof(Elem));  
        if(no == NULL)  
            return 0;  
        no->dados = al;  
        no->prox = (*pi);  
        *pi = no;  
        return 1;  
    }
```



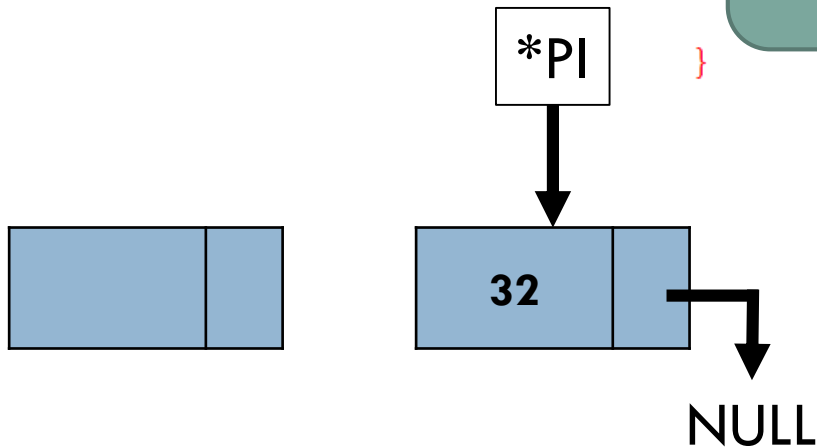
Exemplo de Pilha: Inserção

107

AL: 98

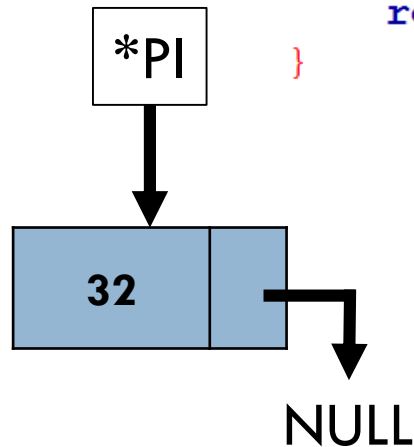
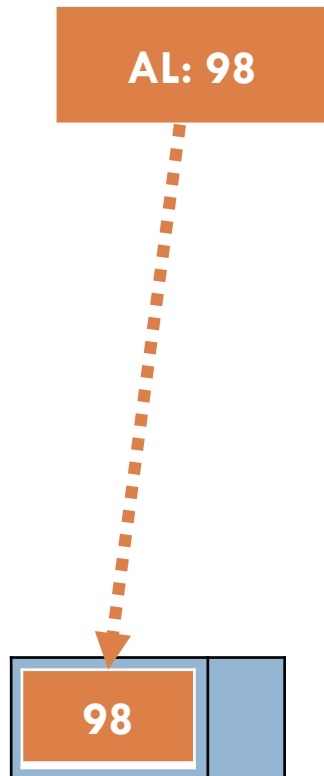
```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    → if(no == NULL)
```

NO == NULL : NÃO ALOCADO.



Exemplo de Pilha: Inserção

108



```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    → no->dados = al;  
    no->prox = (*pi);  
    *pi = no;  
    return 1;  
}
```

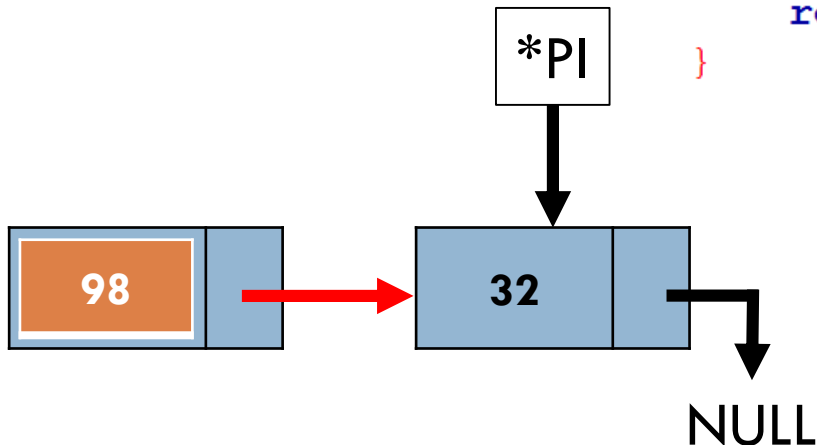
Exemplo de Pilha: Inserção

109

AL: 98

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = (*pi);  
    *pi = no;  
    return 1;  
}
```

SEMPRE APONTA PARA O INICIO DA PILHA.

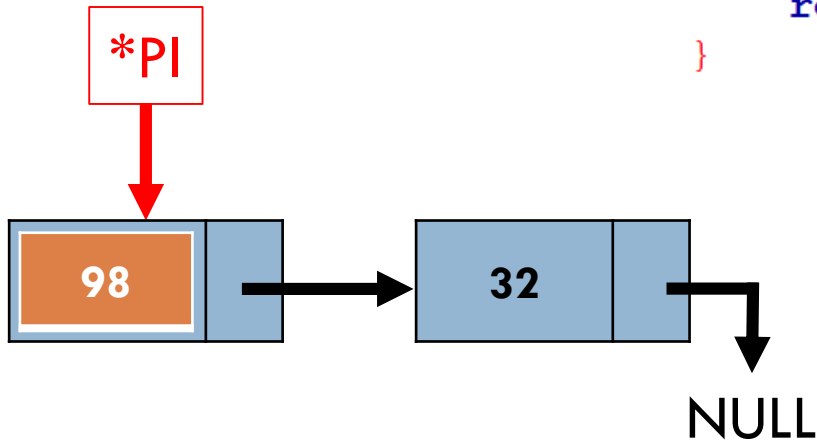


Exemplo de Pilha: Inserção

110

AL: 98

```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = (*pi);  
    → *pi = no;  
    return 1;  
}
```

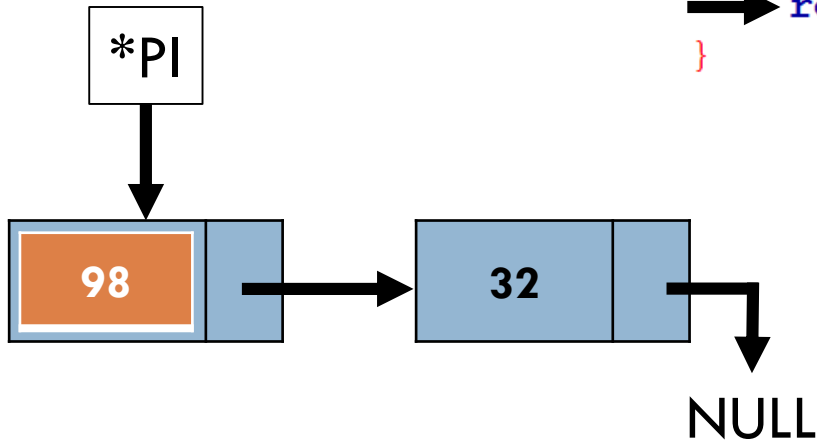


Exemplo de Pilha: Inserção

111

AL: 98

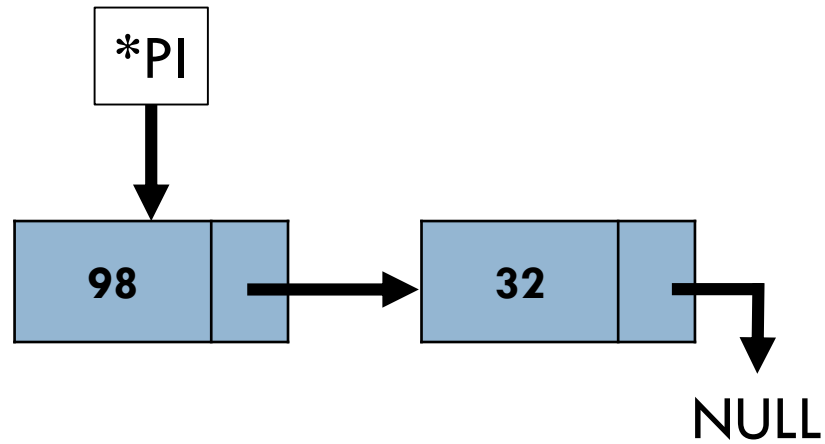
```
int insere_Pilha(Pilha* pi, struct aluno al){  
    if(pi == NULL)  
        return 0;  
    Elem* no;  
    no = (Elem*) malloc(sizeof(Elem));  
    if(no == NULL)  
        return 0;  
    no->dados = al;  
    no->prox = (*pi);  
    *pi = no;  
    return 1;  
}
```



- RETORNA OK;
- FINAL DA INSERÇÃO DO ELEMENTO.

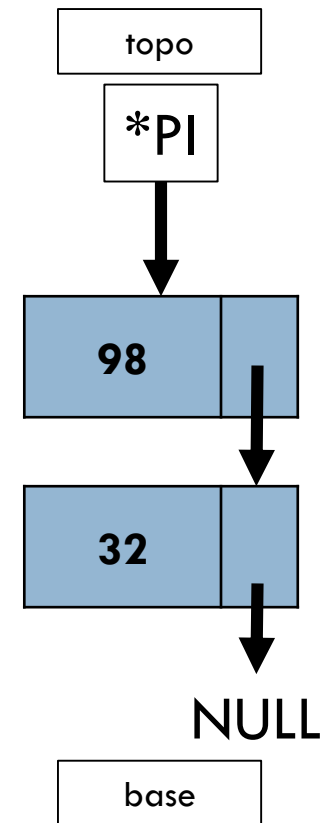
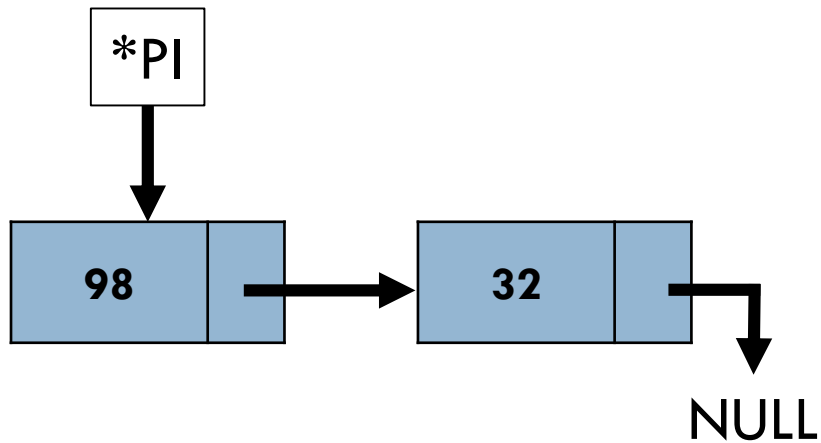
Exemplo de Pilha: Inserção

112



Exemplo de Pilha: Inserção

113



Exemplo de Pilha: Remoção

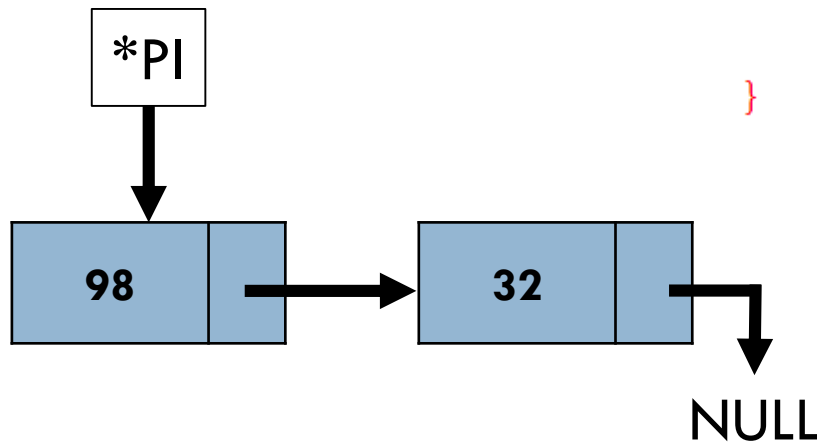
114

```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    return 1;  
}
```

Exemplo de Pilha: Remoção

115

```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    return 1;  
}
```

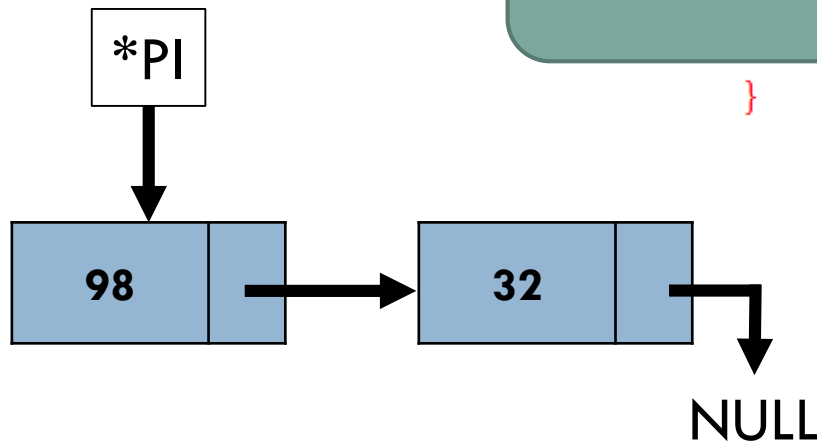


Exemplo de Pilha: Remoção

116

```
int remove_Pilha(Pilha* pi) {  
    → if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)
```

PILHA == NULL : NÃO ALOCADA.

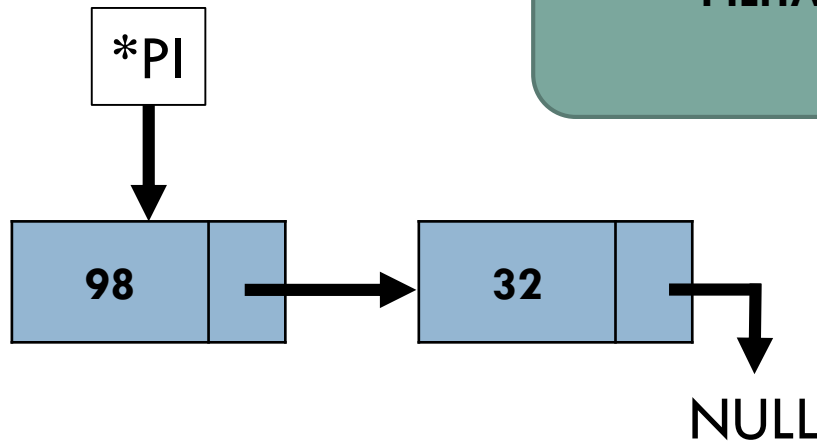


Exemplo de Pilha: Remoção

117

```
int remove_Pilha(Pilha* pi){  
    if(pi == NULL)  
        return 0;  
    → if((*pi) == NULL)  
        return 0;
```

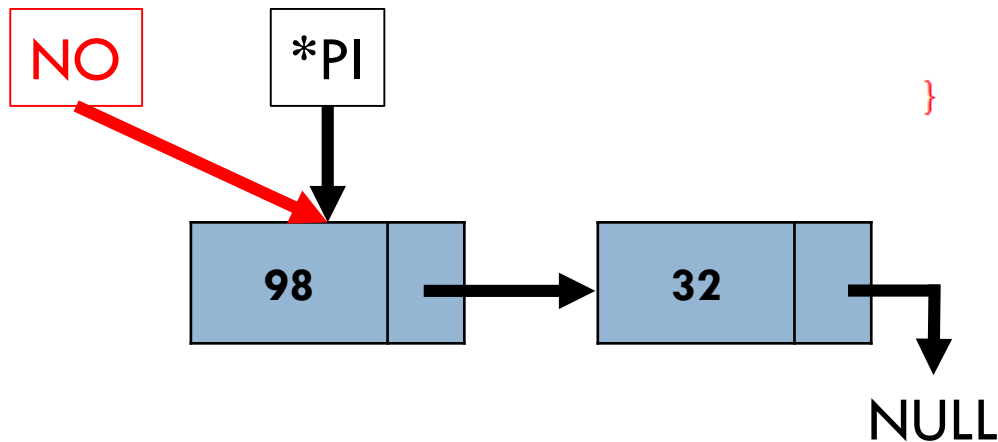
***PILHA == NULL : PILHA VAZIA.**



Exemplo de Pilha: Remoção

118

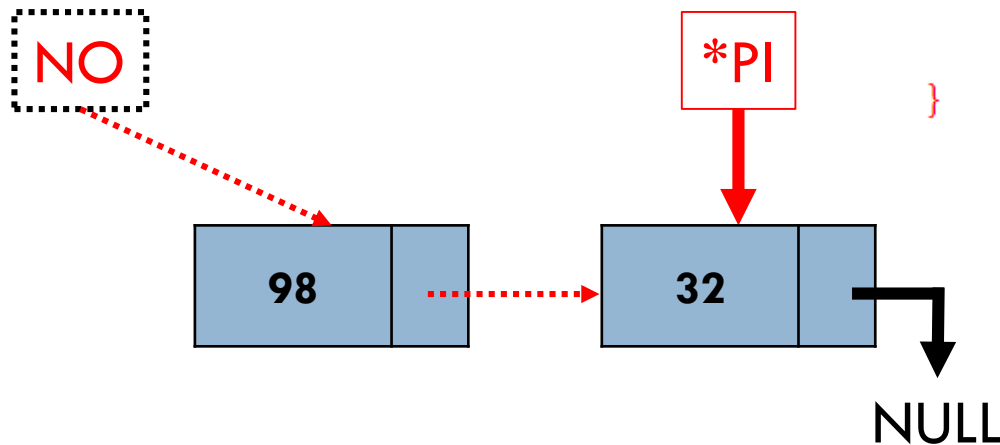
```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    → Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    return 1;  
}
```



Exemplo de Pilha: Remoção

119

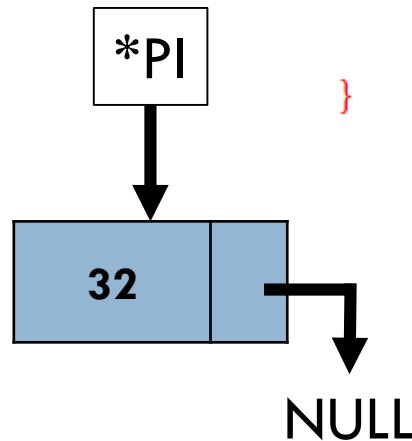
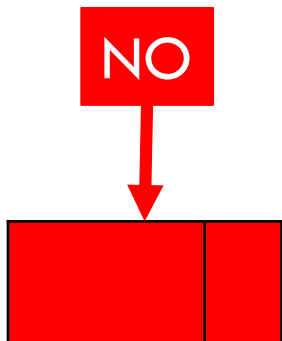
```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    → *pi = no->prox;  
    free(no);  
    return 1;  
}
```



Exemplo de Pilha: Remoção

120

```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    return 1;  
}
```

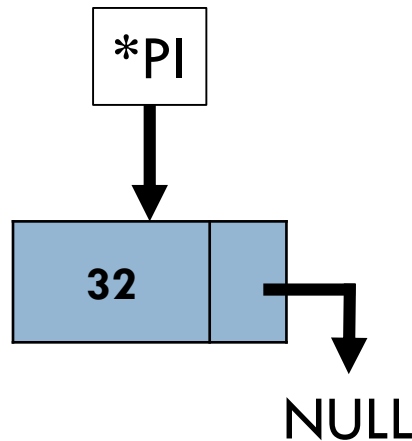


Exemplo de Pilha: Remoção

121

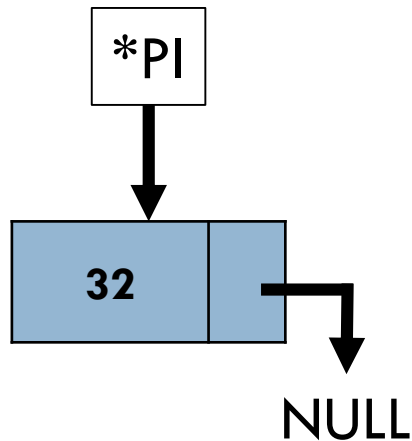
- RETORNA OK;
- FINAL DA REMOÇÃO DO ELEMENTO.

```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    return 1;  
}
```



Exemplo de Pilha: Remoção

122



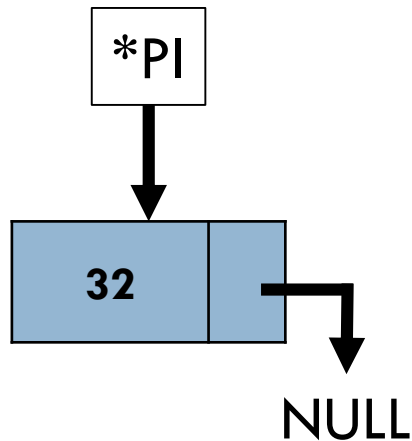
```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    return 1;  
}
```

Exemplo de Pilha: Remoção

123

```
int remove_Pilha(Pilha* pi) {  
    → if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)
```

PILHA == NULL : NÃO ALOCADA.

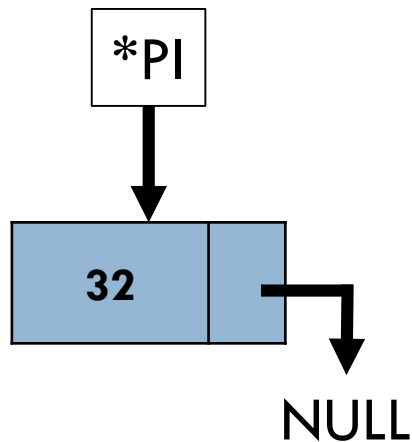


```
}
```

Exemplo de Pilha: Remoção

124

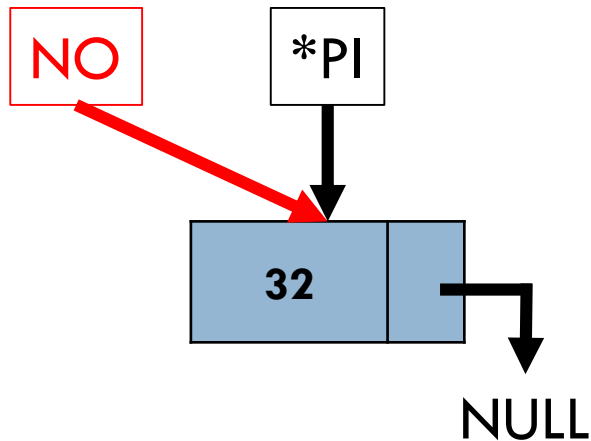
```
int remove_Pilha(Pilha* pi){  
    if(pi == NULL)  
        return 0;  
    → if((*pi) == NULL)  
        return 0;
```



***PILHA == NULL : PILHA VAZIA.**

Exemplo de Pilha: Remoção

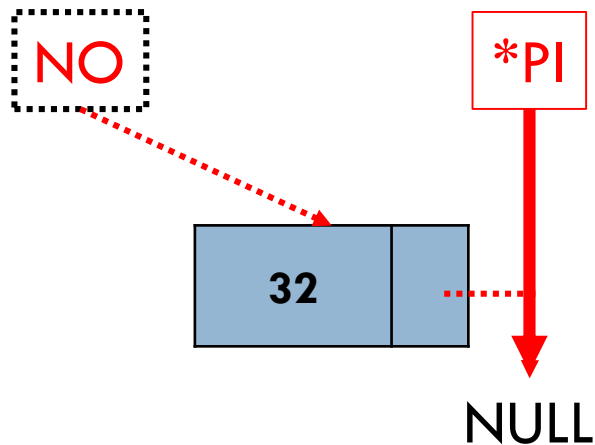
125



```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    → Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    return 1;  
}
```

Exemplo de Pilha: Remoção

126

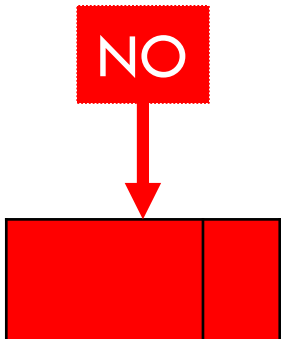


```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    → *pi = no->prox;  
    free(no);  
    return 1;  
}
```

Exemplo de Pilha: Remoção

127

```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    *pi = no->prox;  
    → free(no);  
    return 1;  
}
```



Exemplo de Pilha: Remoção

128

- RETORNA OK;
- FINAL DA REMOÇÃO DO ELEMENTO.



```
int remove_Pilha(Pilha* pi) {  
    if(pi == NULL)  
        return 0;  
    if((*pi) == NULL)  
        return 0;  
    Elem *no = *pi;  
    *pi = no->prox;  
    free(no);  
    → return 1;  
}
```


Pilha Estática

129

- ❑ Armazena em vetores.
- ❑ Utiliza índice para conhecer o topo da pilha.
- ❑ Simples de trabalhar, utilizando sempre o último **índice disponível como topo.**

Pilha: Exercícios

130

- ❑ Crie uma função de consulta para a Pilha do exemplo. **A consulta só é realizada no elemento do topo.**
- ❑ Adapte a **lista estática** para funcionar como uma **pilha**.

Deque

131

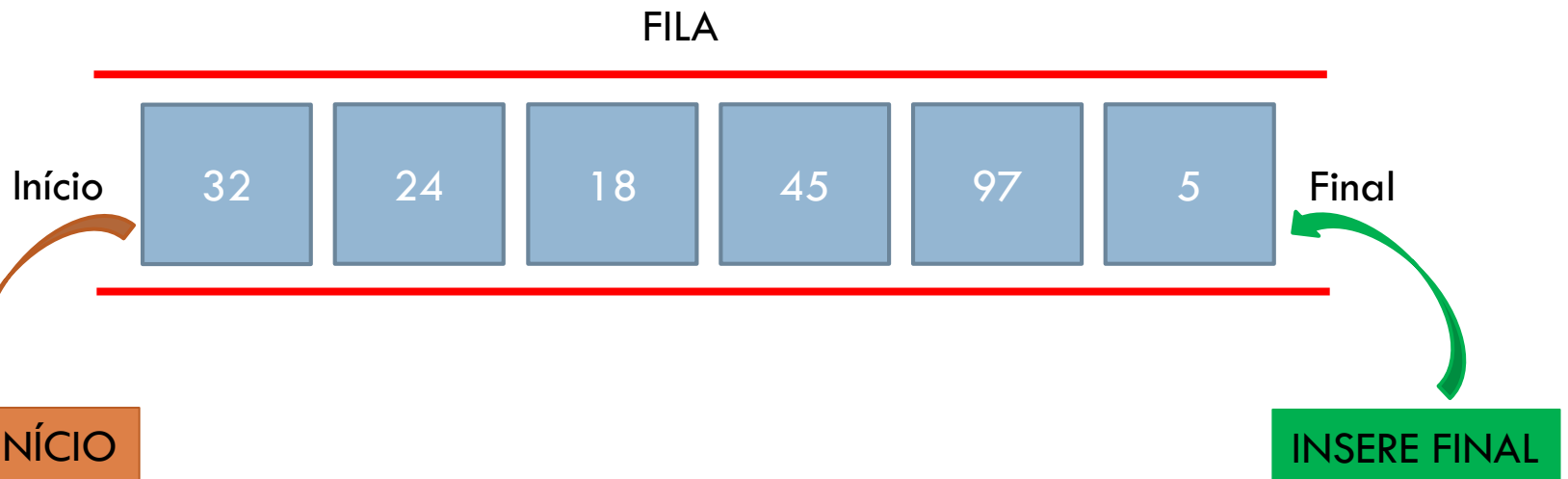
- Tipo especial de **fila e pilha**.
- Permite inserção e remoção em ambas as extremidades.
- Double Ended QUEUE: fila com duas saídas.

- Aplicações:
 - ▣ Verificação de palíndromo.
 - ▣ Escalonamento de processos.

Dequeues

132

□ FILA:

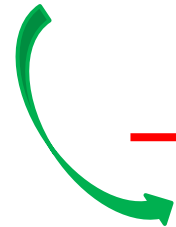


Dequeues

133

□ PILHA.

INSERE INÍCIO

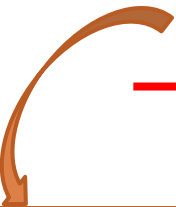


PILHA

Início



Final

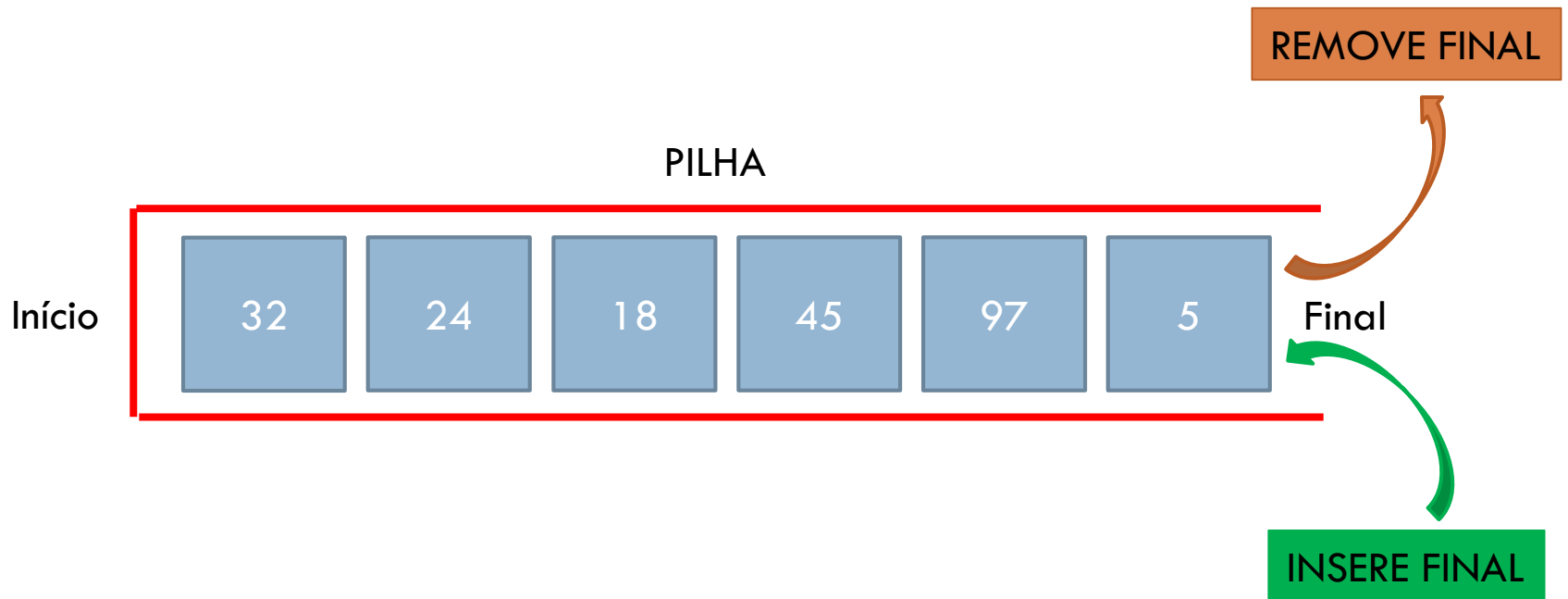


REMOVE INÍCIO

Dequeues

134

□ PILHA.



Deques

135

- Inserção e remoção de ambas as extremidades.

