FILA, PILHA E DEQUE

Prof. Muriel Mazzetto

Estrutura de Dados

Listas

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

Listas

- 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

- 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

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

Fila

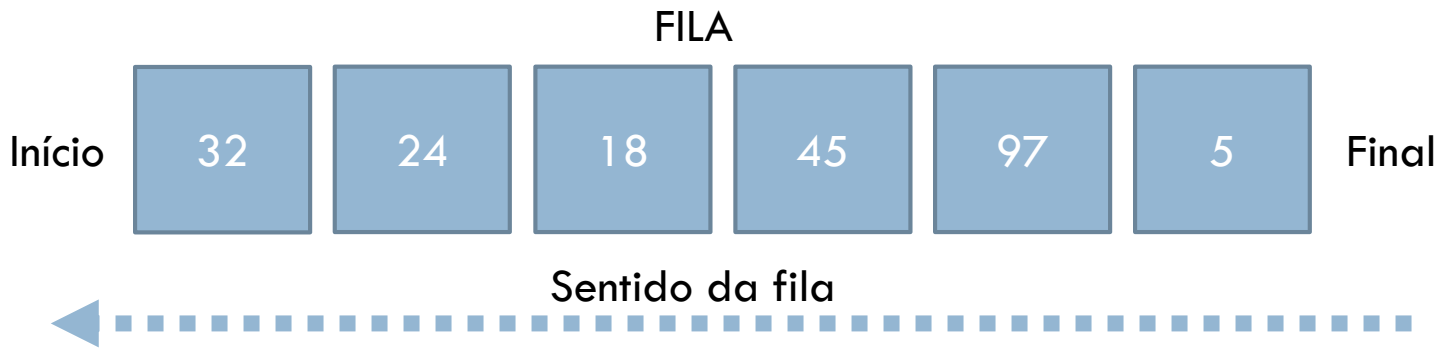
- Tipo especial de lista.
- Sentido de inserção e remoção.

Fila

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

Fila

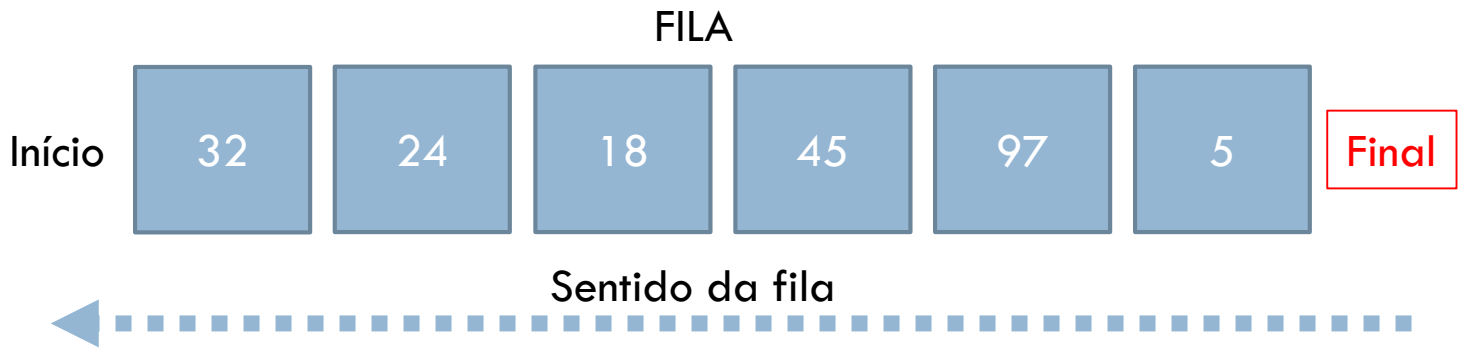
- Tipo especial de lista.
- Sentido de inserção e remoção.



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

Fila

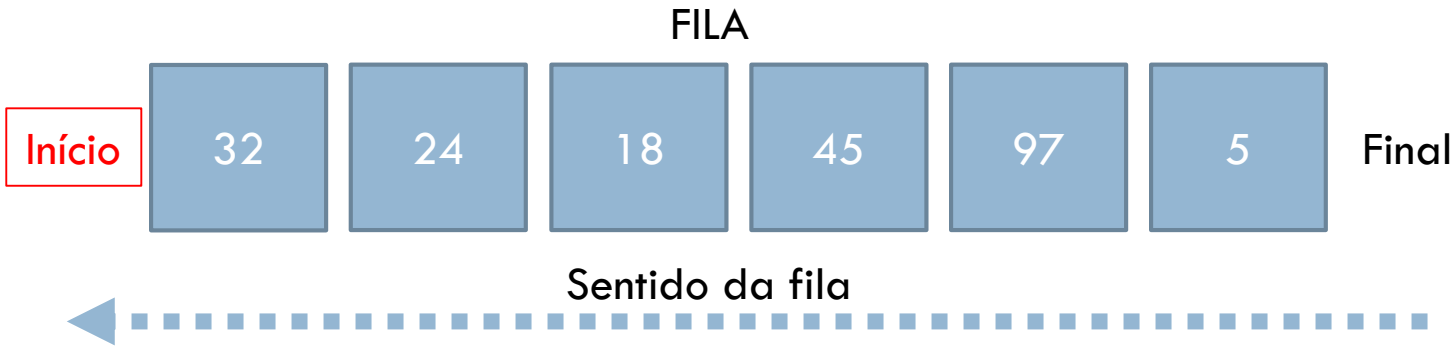
- Tipo especial de lista.
- Sentido de inserção e remoção.



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

Fila

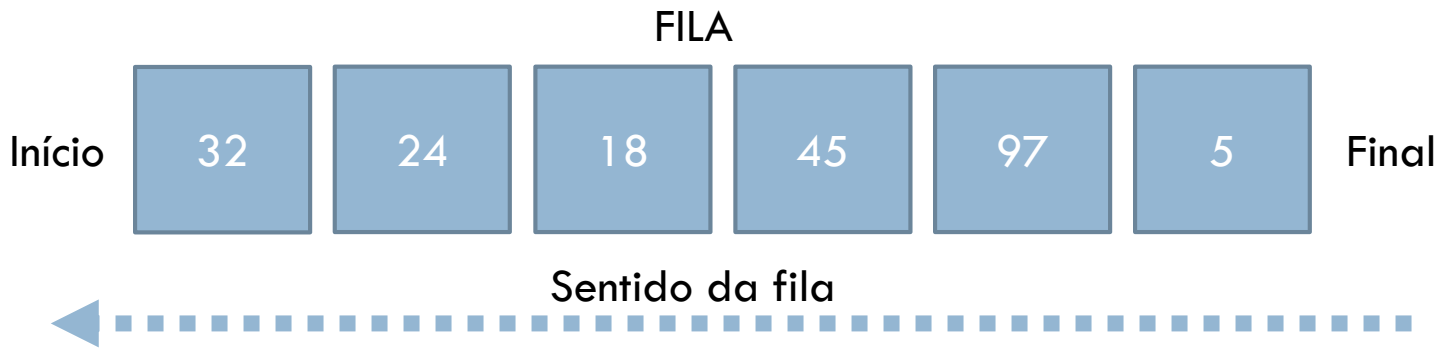
- Tipo especial de lista.
- Sentido de inserção e remoção.



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

Fila

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



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

Fila

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

Fila

- 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

- 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

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

Exemplo de Fila: Definição de tipos


```c
//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


```c
//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


```c
//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


```c
//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;
};
```

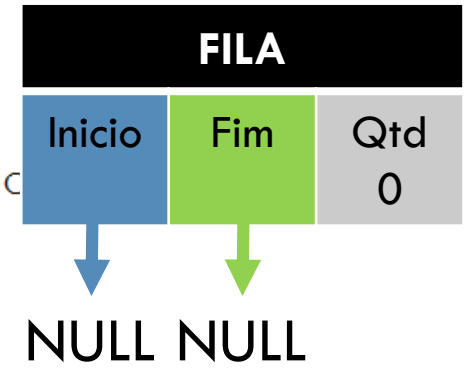
Exemplo de Fila: Criar/Liberar


```
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


```
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
            free(no);
        }
        free(fi);
    }
}
```

INICIALIZAR TODA A ESTRUTURA DE DESCRITOR.



Exemplo de Fila: Criar/Liberar


```
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);
    }
}
```

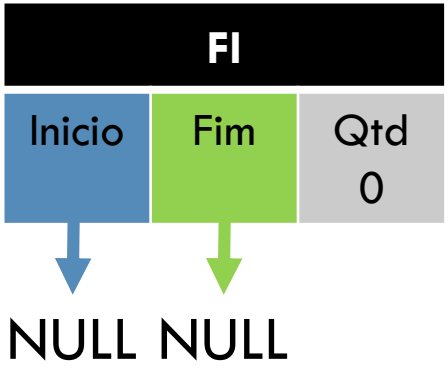
DESALOCAR SEMPRE O PRIMEIRO ELEMENTO, ATÉ QUE A FILA FIQUE VAZIA.

Exemplo de Fila: Inserção


```
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

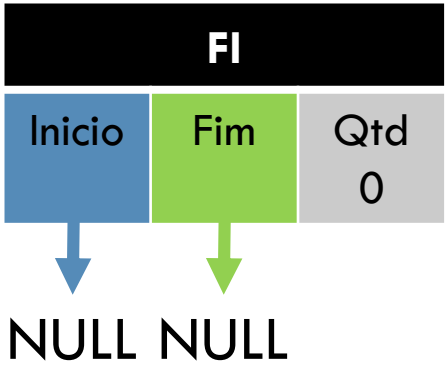
AL: 32




```c
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

AL: 32

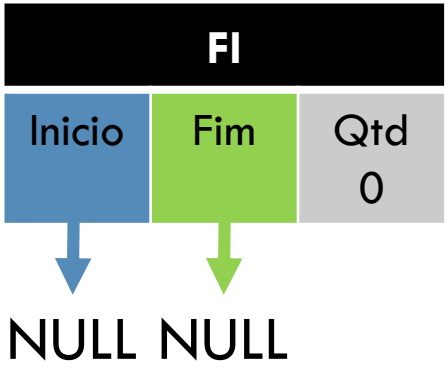



```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL)
        fi->inicio = no;
    else
        fi->fim->prox = no;
    fi->fim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

- **FILA == NULL : NÃO ALOCADA.**

Exemplo de Fila: Inserção

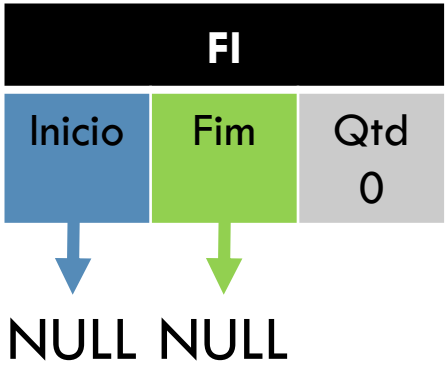
AL: 32



```
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

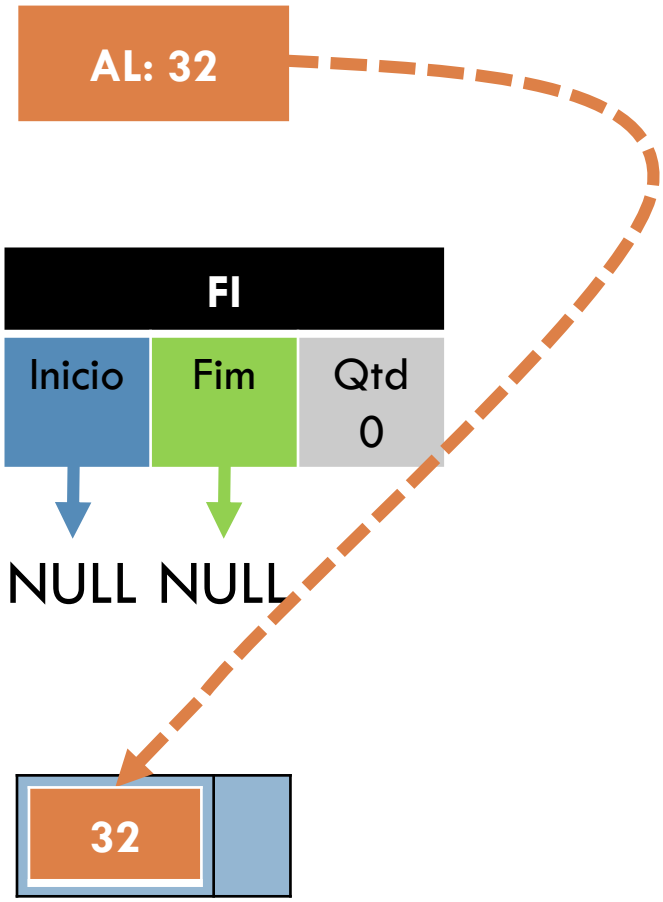
AL: 32



```
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.**

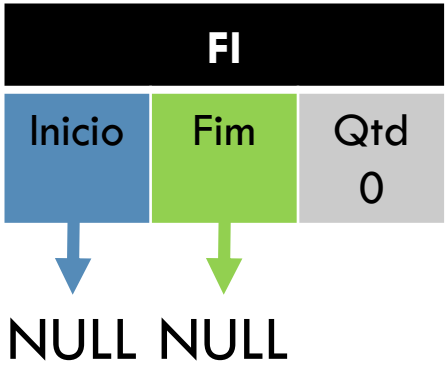
Exemplo de Fila: Inserção

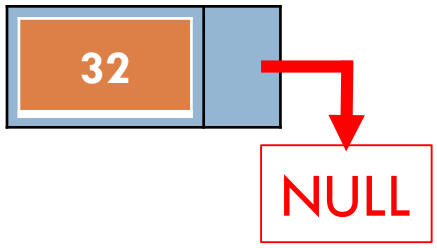



```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL)//fila vazia
        fi->inicio = no;
    else
        fi->fim->prox =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

Exemplo de Fila: Inserção

AL: 32

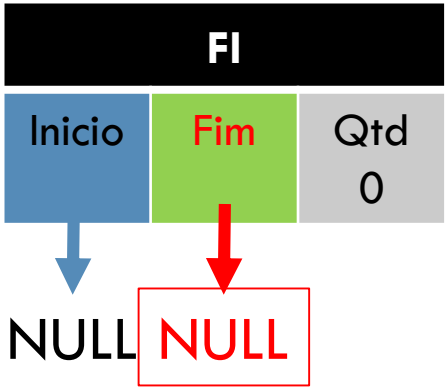


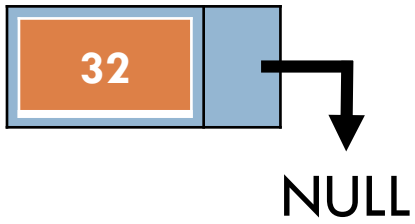



```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL)//fila vazia
        fi->inicio = no;
    else
        fi->fim->prox =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

Exemplo de Fila: Inserção

AL: 32

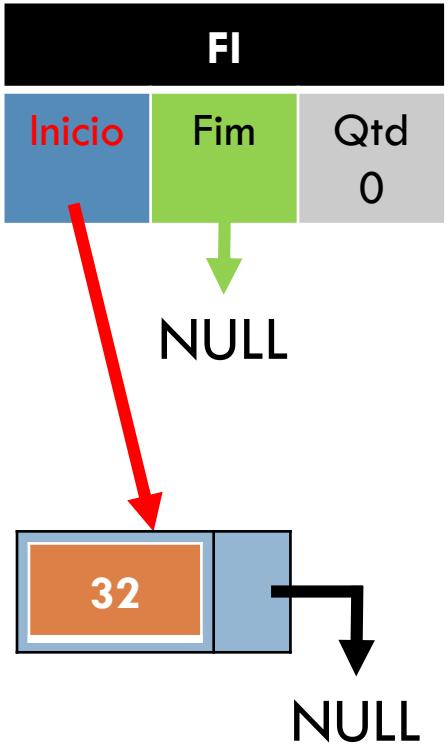





```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL)//fila vazia
        fi->inicio = no;
    else
        fi->fim->prox =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

Exemplo de Fila: Inserção

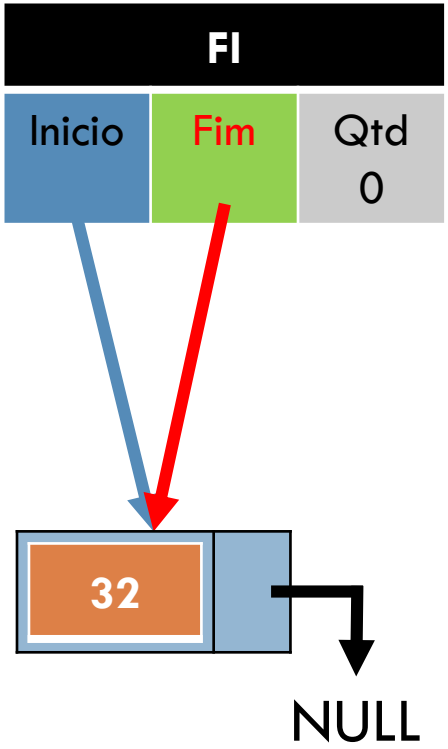
AL: 32




```
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

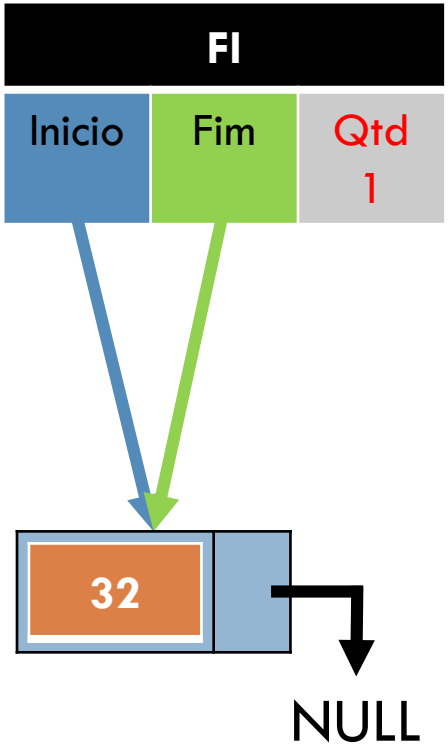
AL: 32




```
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

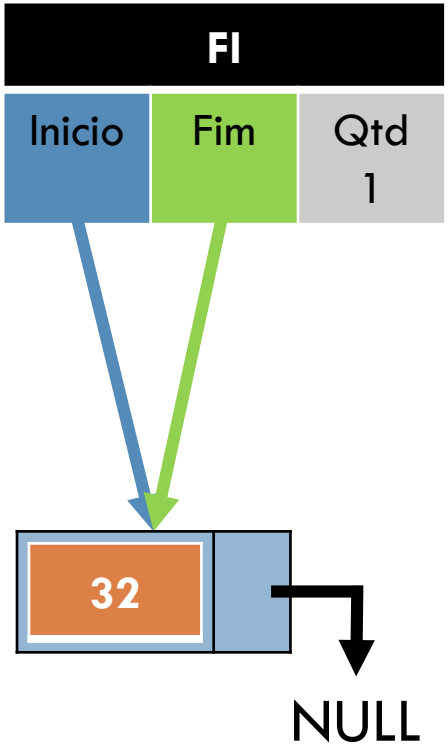
AL: 32




```
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

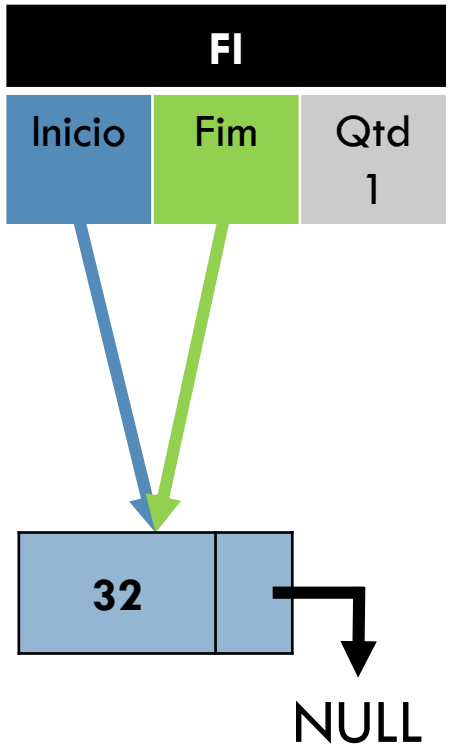
AL: 32




```
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;
}
```

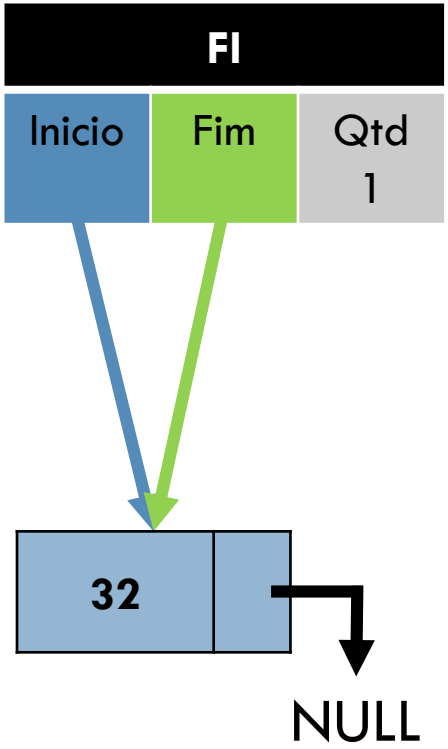
- **RETORNA OK;**
- **FINAL DA INSERÇÃO DO ELEMENTO.**

Exemplo de Fila: Inserção



Exemplo de Fila: Inserção

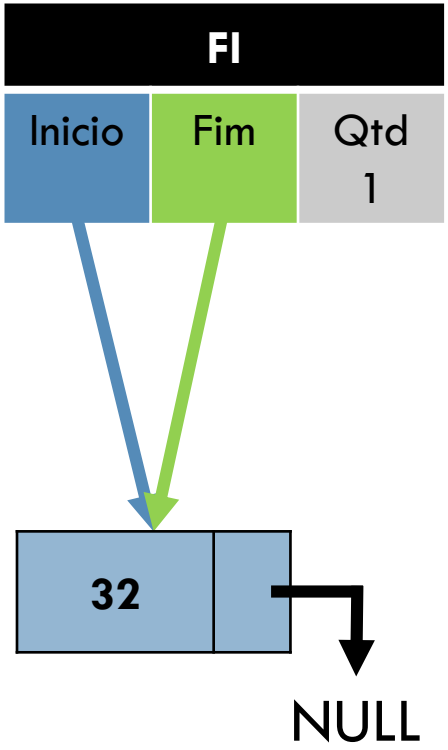
AL: 98




```
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

AL: 98

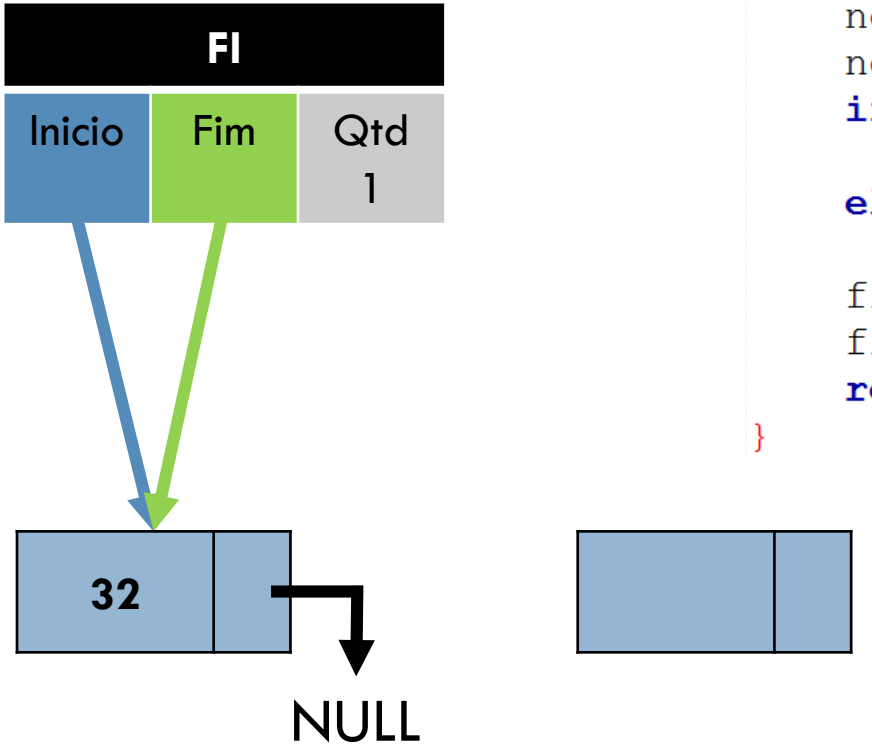



```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL)
        fi->inicio = no;
    else
        fi->fim->prox = no;
    fi->fim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

- **FILA == NULL : NÃO ALOCADA.**

Exemplo de Fila: Inserção

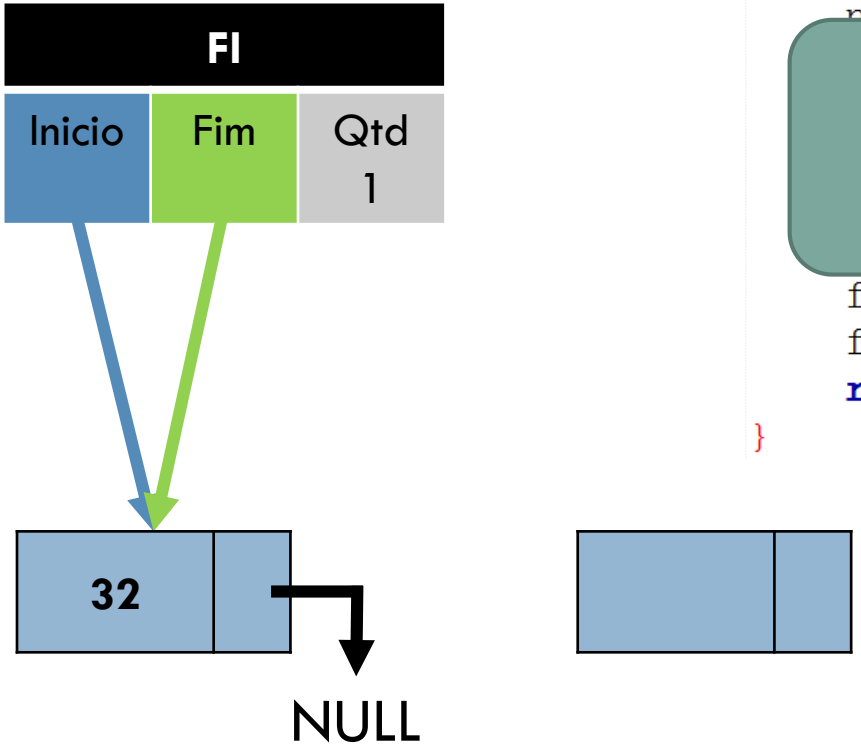
AL: 98




```
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

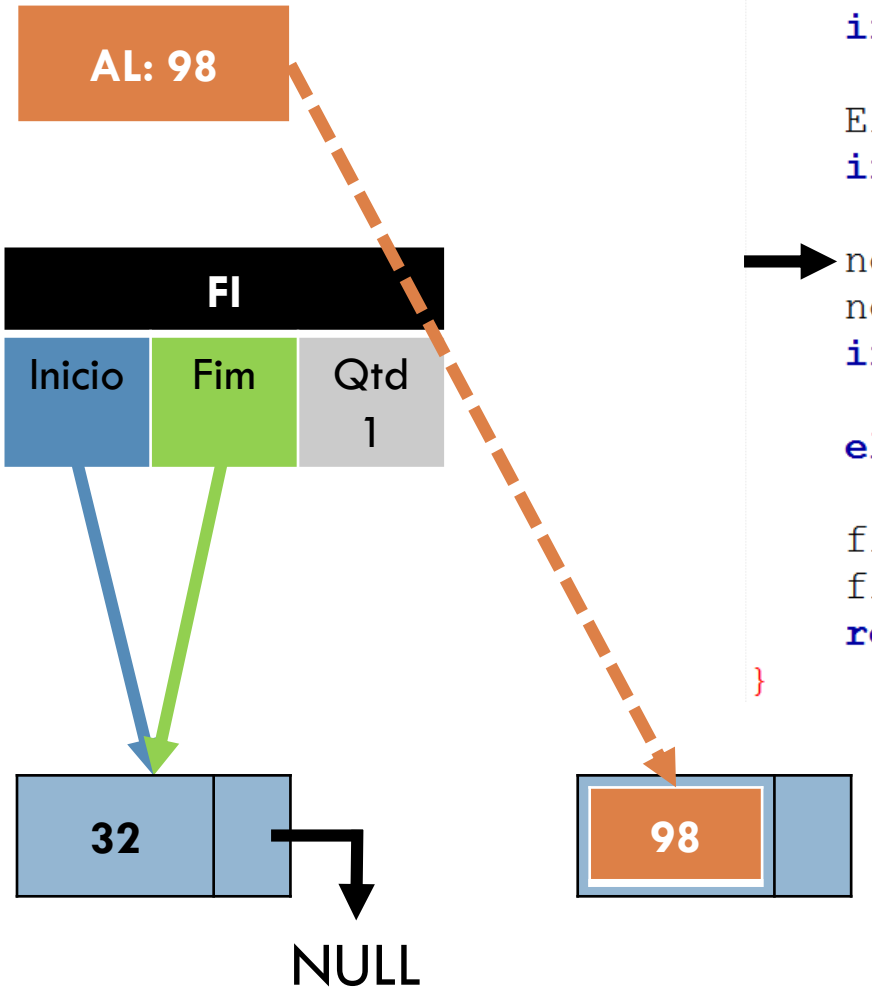
AL: 98




```
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.**

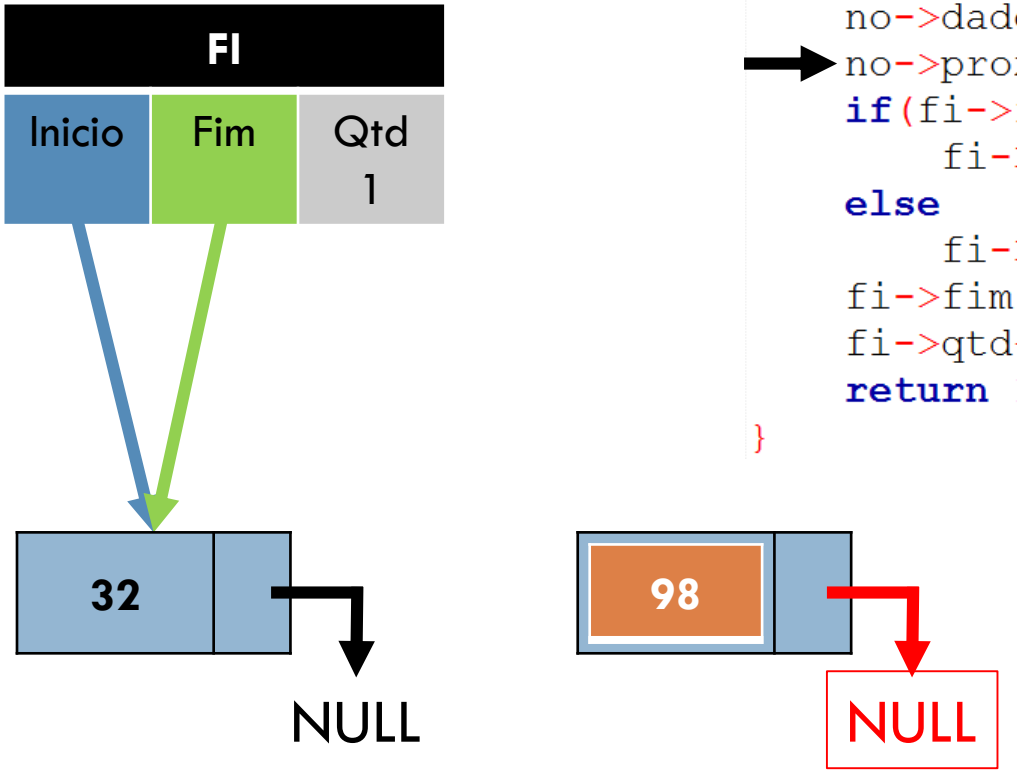
Exemplo de Fila: Inserção




```
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

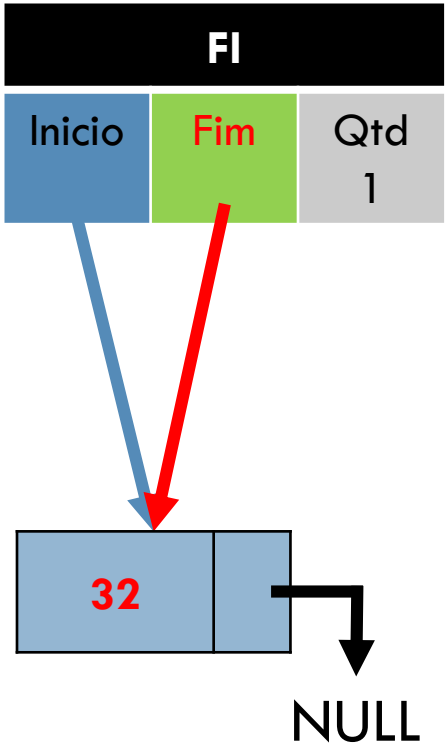
AL: 98

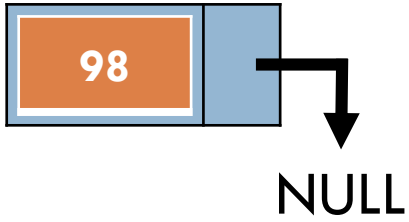



```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL)//fila vazia
        fi->inicio = no;
    else
        fi->fim->prox =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

Exemplo de Fila: Inserção

AL: 98

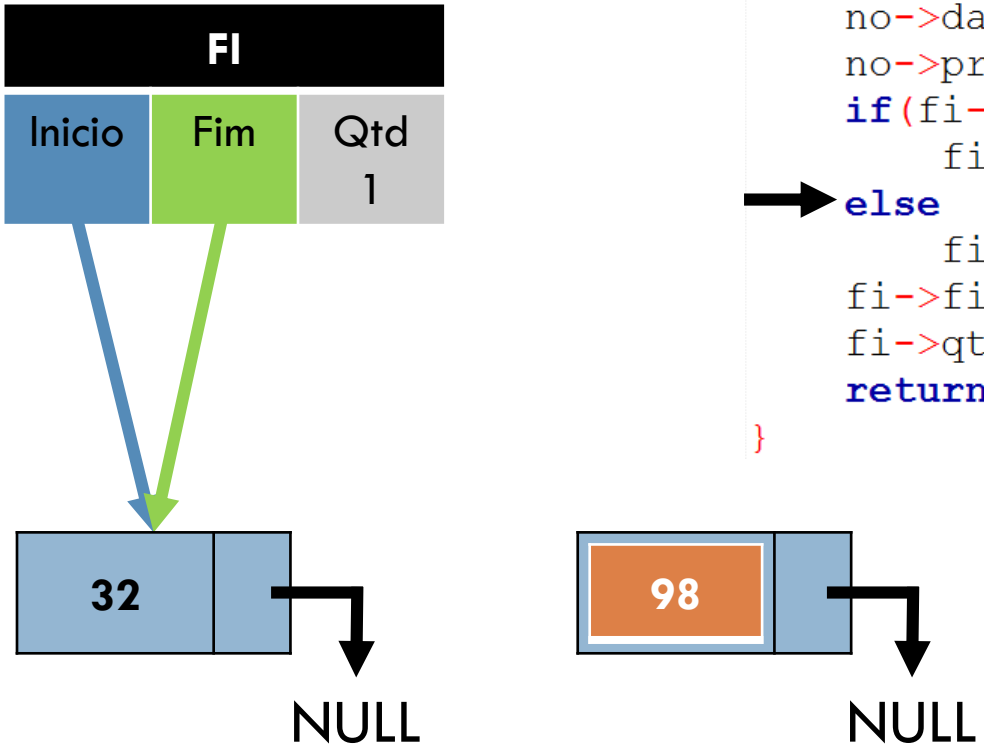





```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL)//fila vazia
        fi->inicio = no;
    else
        fi->fim->prox =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

Exemplo de Fila: Inserção

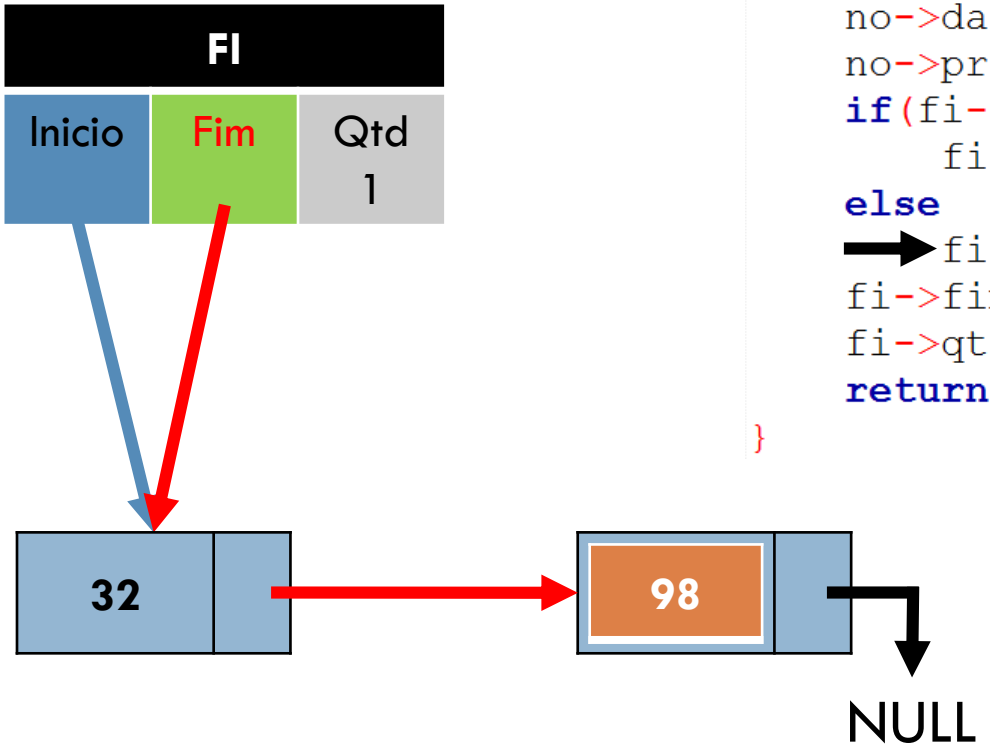
AL: 98




```
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

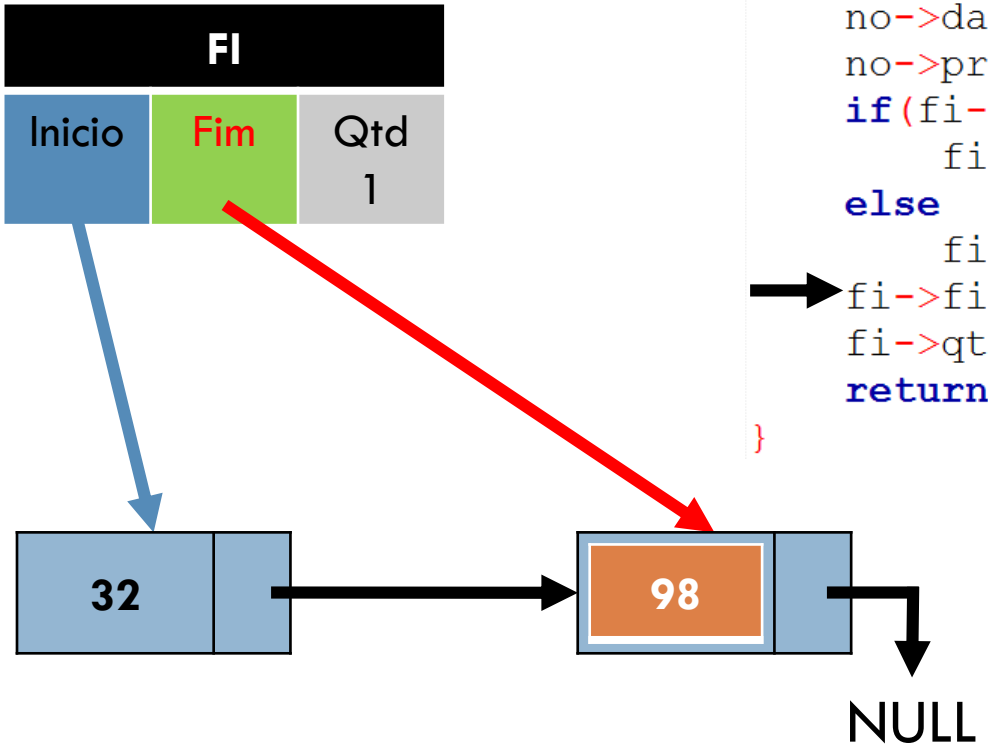
AL: 98




```
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

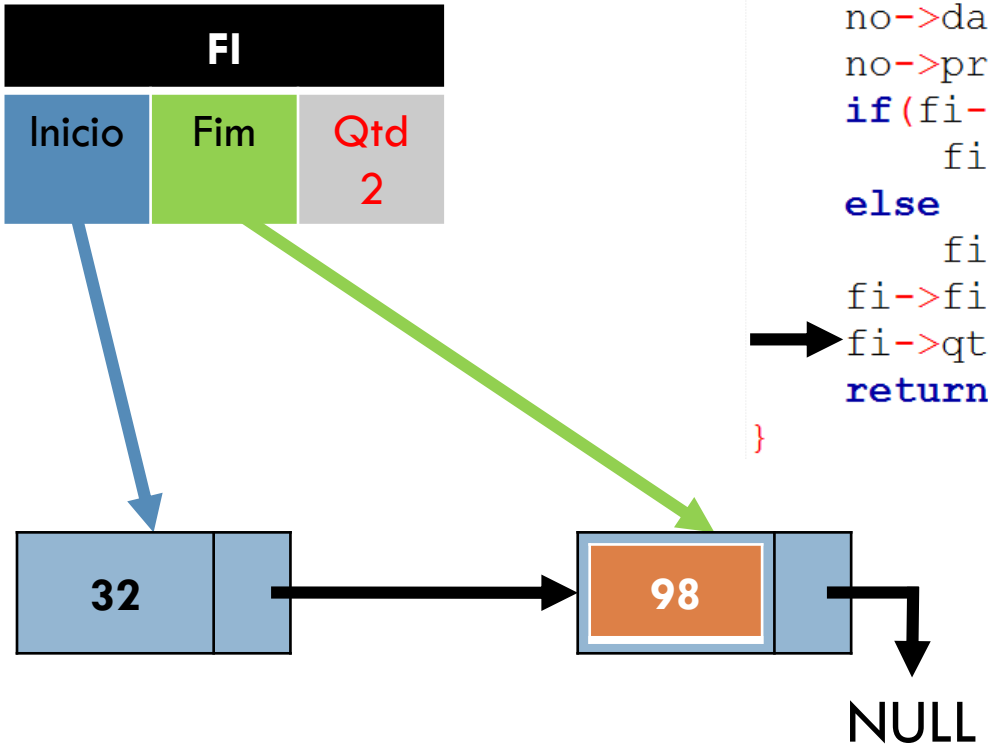
AL: 98




```
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

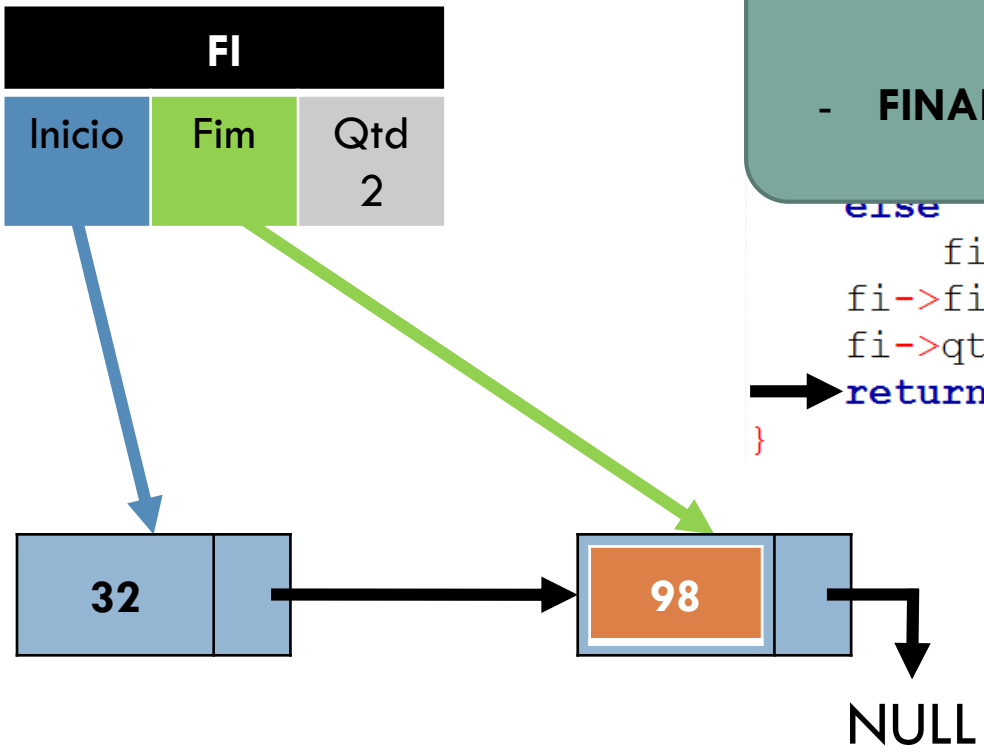
AL: 98




```
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

AL: 98

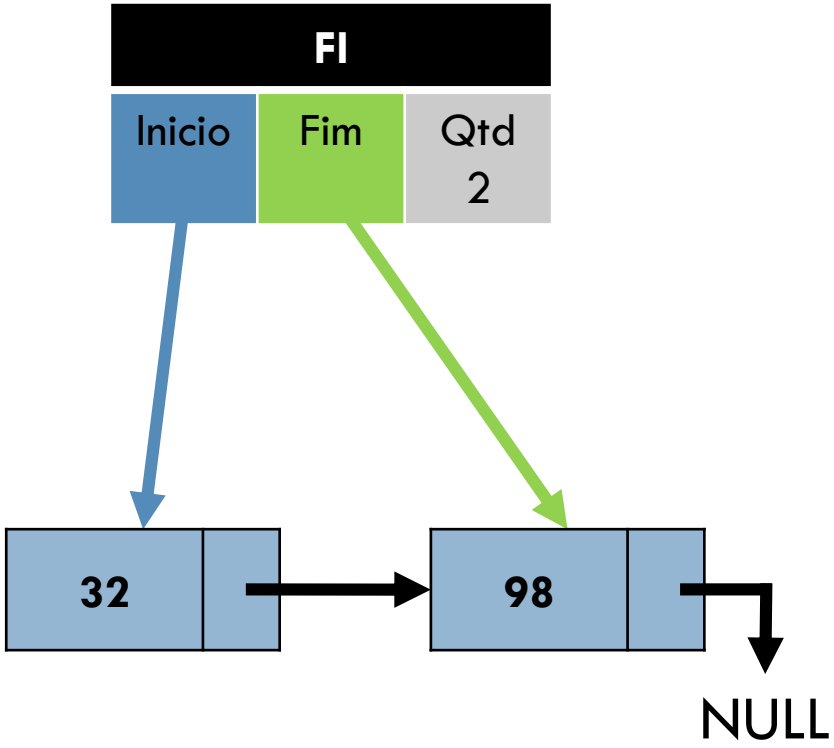



```
int insere_Fila(Fila* fi, struct aluno al){
    if(fi == NULL)
        return 0;
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
```

- **RETORNA OK;**
- **FINAL DA INSERÇÃO DO ELEMENTO.**

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

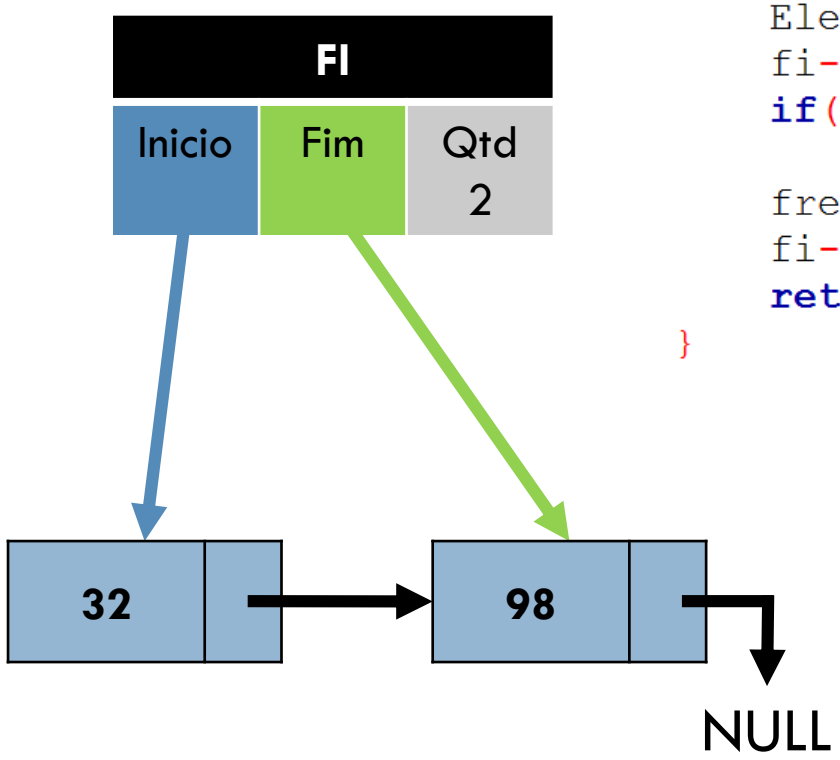
Exemplo de Fila: Inserção



Exemplo de Fila: Remoção


```
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

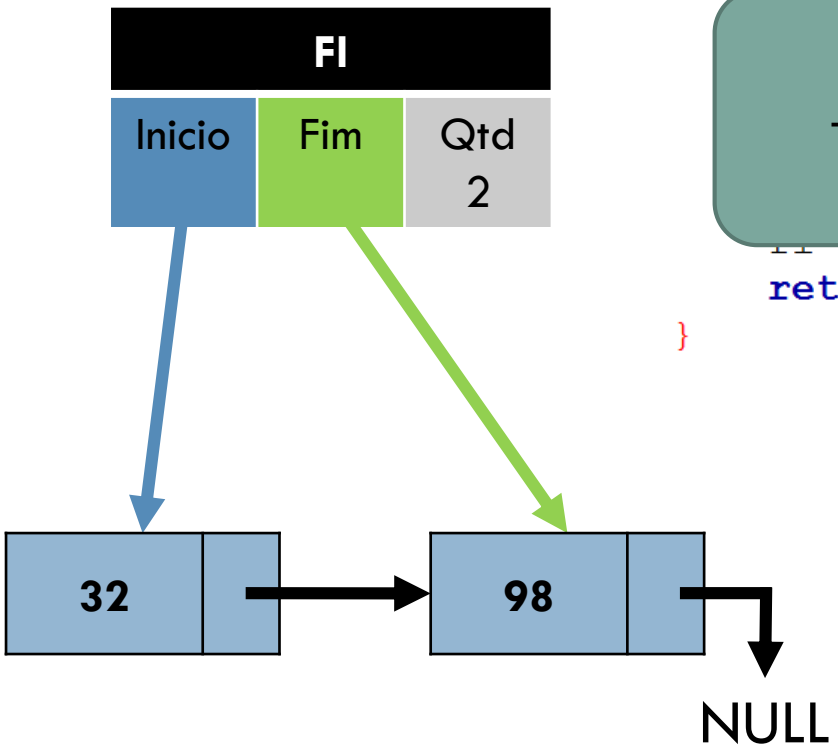



```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

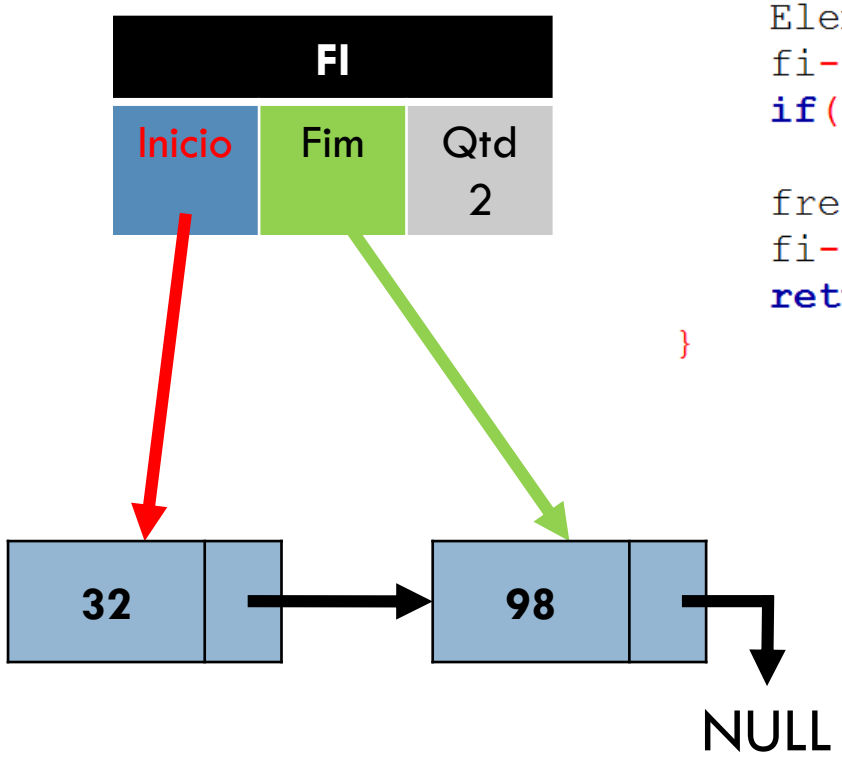
Exemplo de Fila: Remoção


```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    return 1;
}
```

- **FILA == NULL : NÃO ALOCADA.**

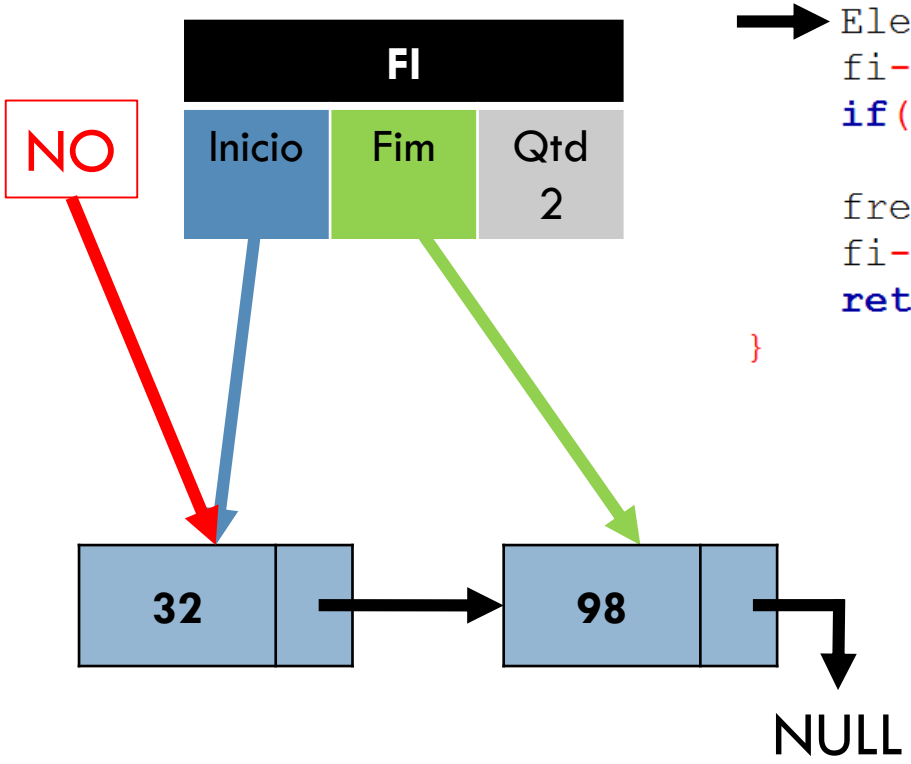


Exemplo de Fila: Remoção



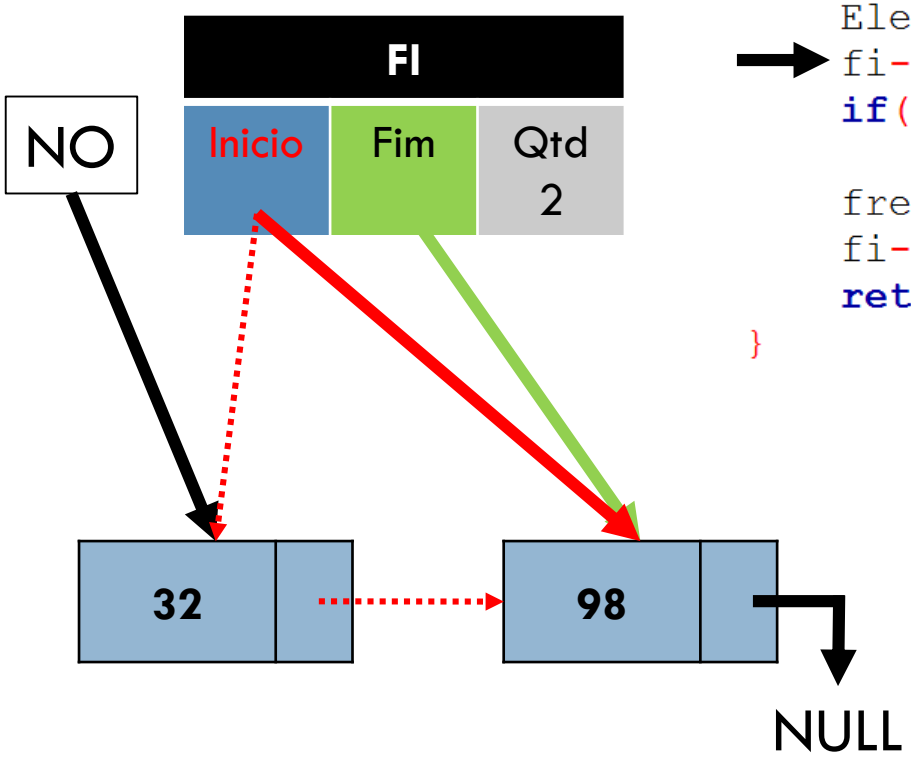

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



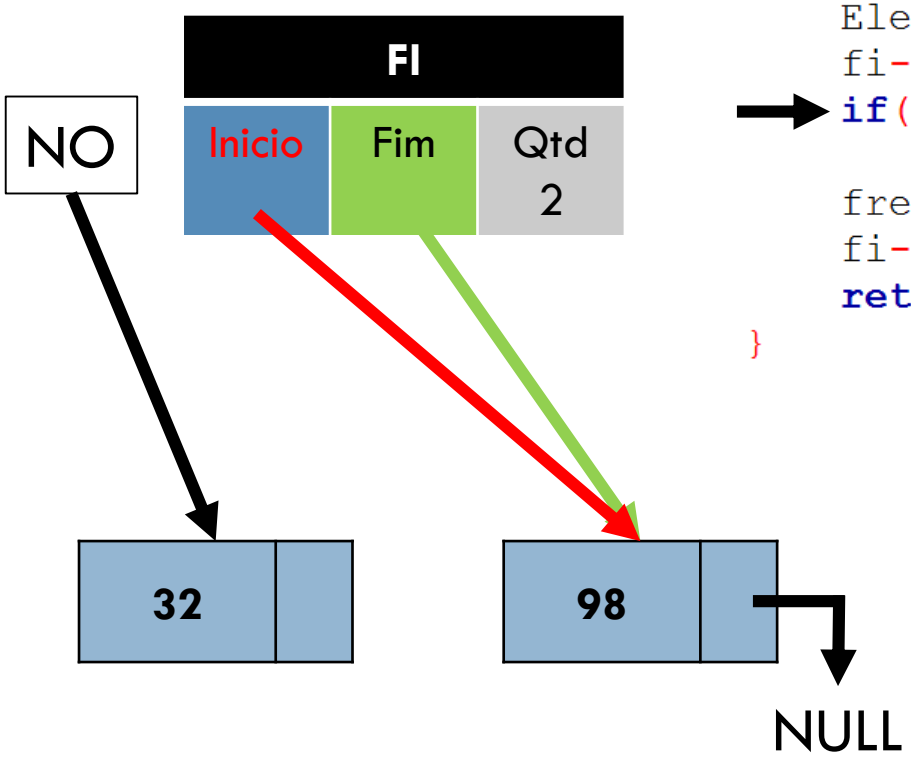

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



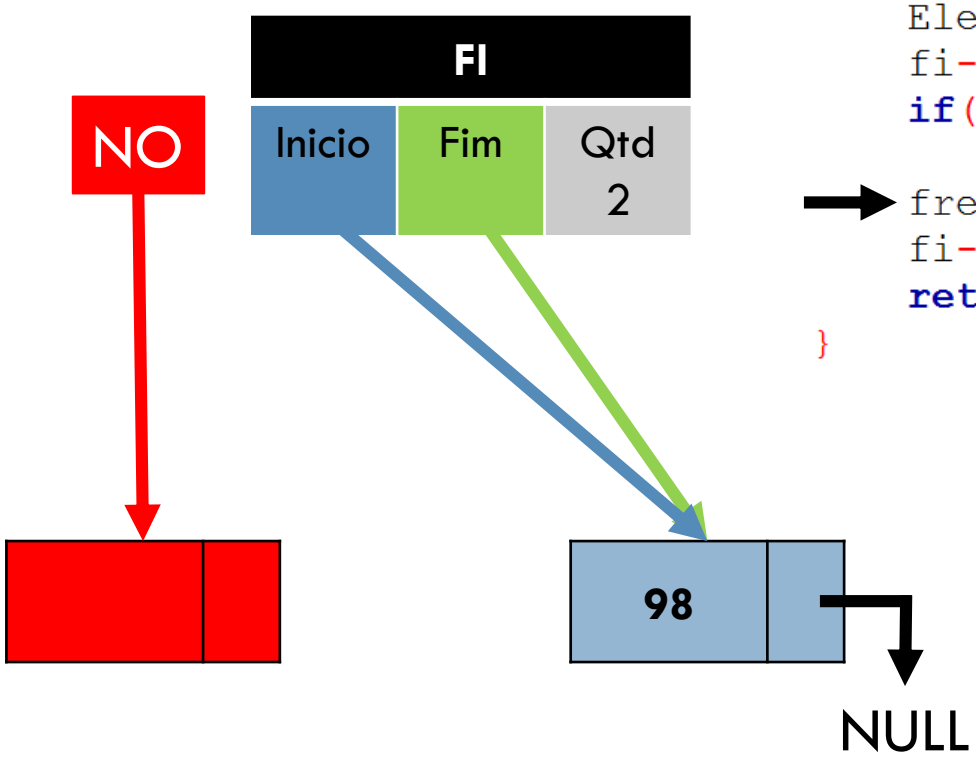

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



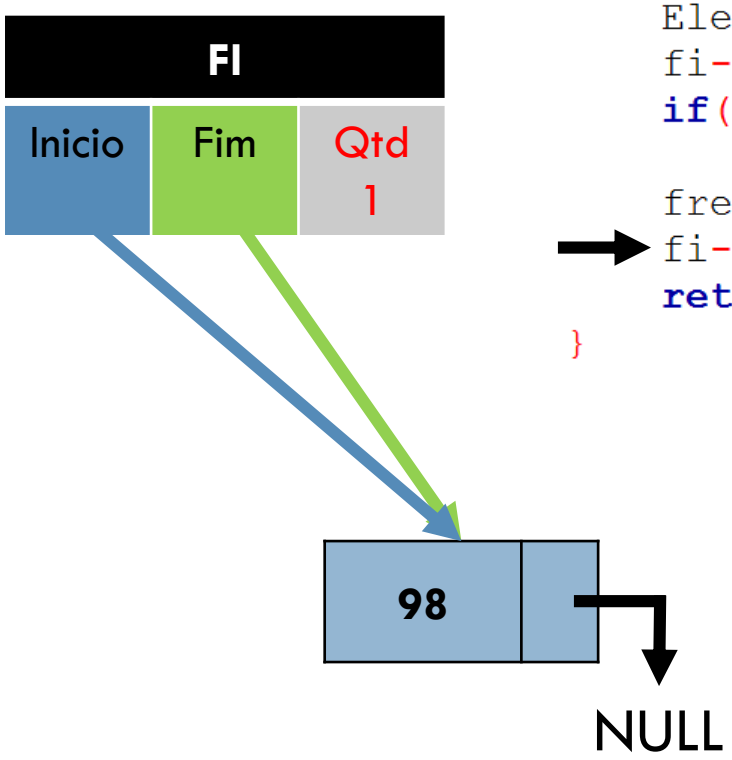

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



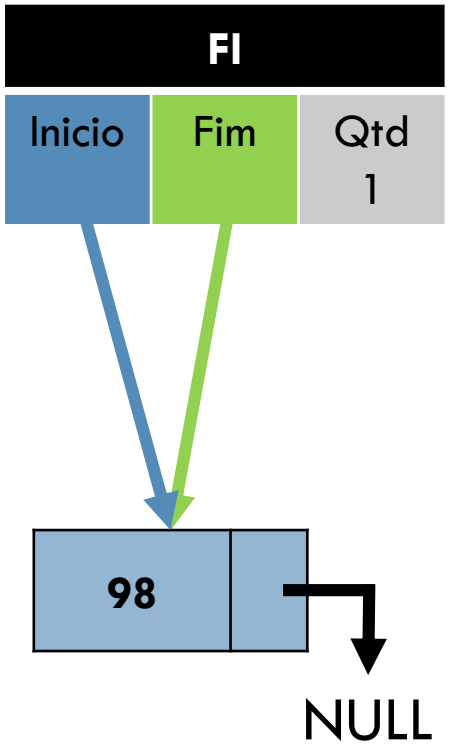

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção




```
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;
}
```

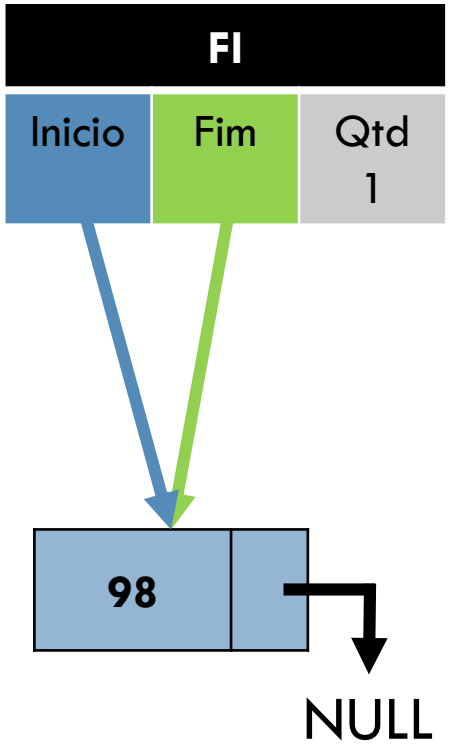
Exemplo de Fila: Remoção



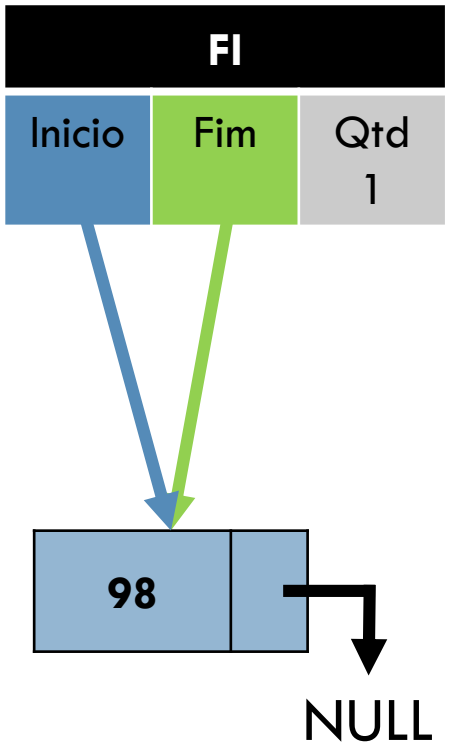

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

- **RETORNA OK;**
- **FINAL DA REMOÇÃO DO ELEMENTO.**

Exemplo de Fila: Remoção

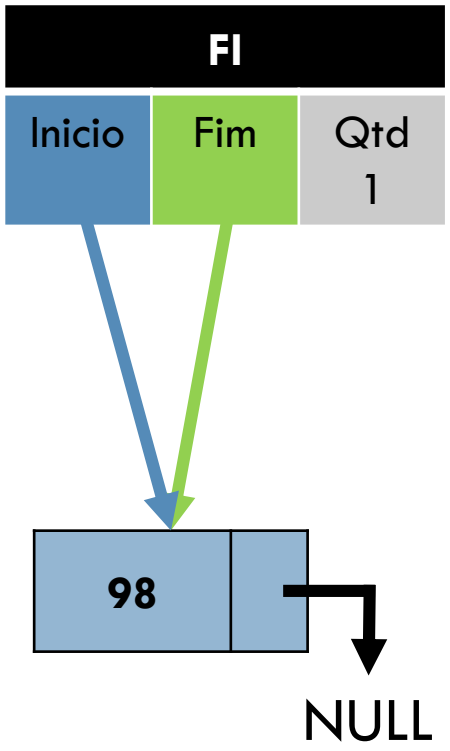


Exemplo de Fila: Remoção




```
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;
}
```

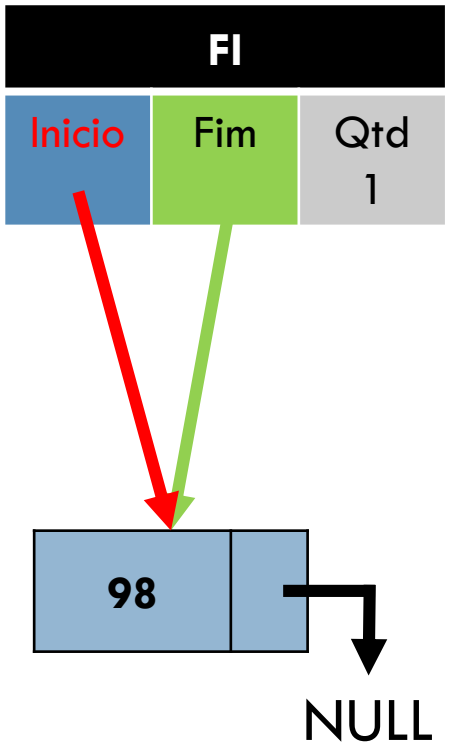
Exemplo de Fila: Remoção




```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    return 1;
}
```

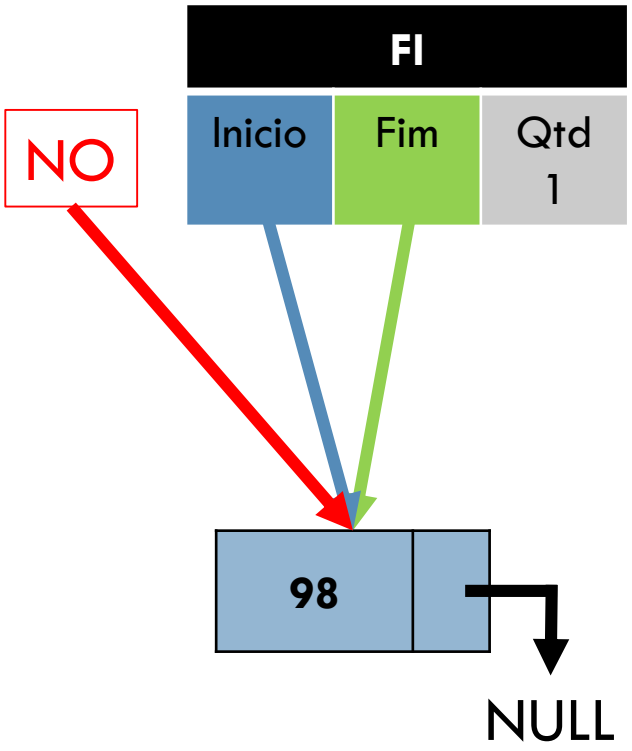
- **FILA == NULL : NÃO ALOCADA.**

Exemplo de Fila: Remoção



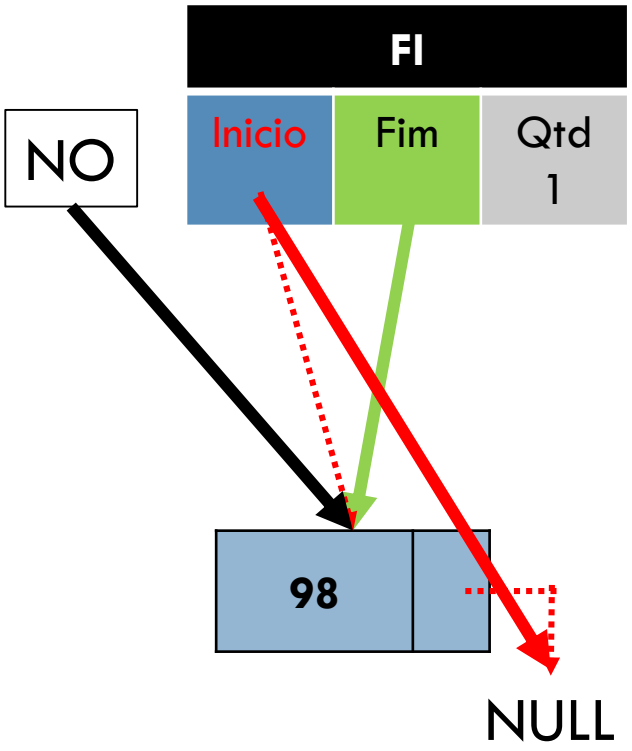

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



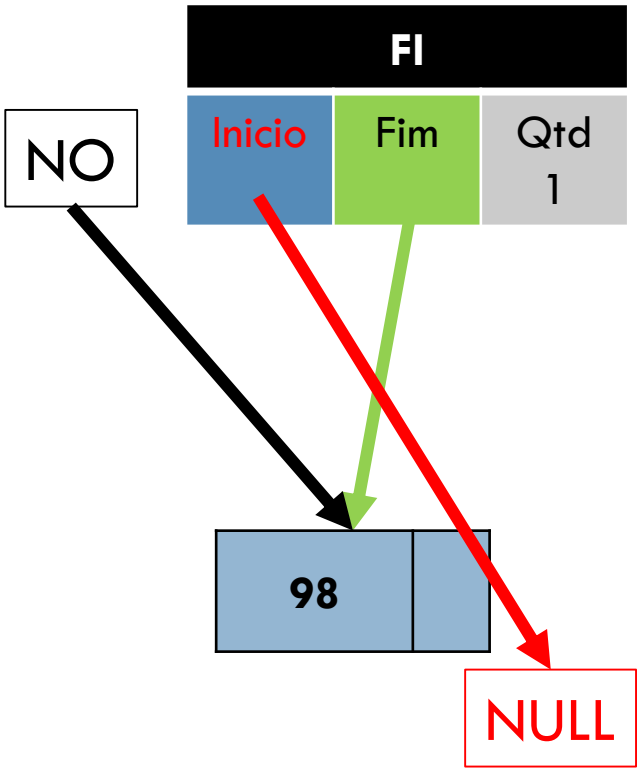

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



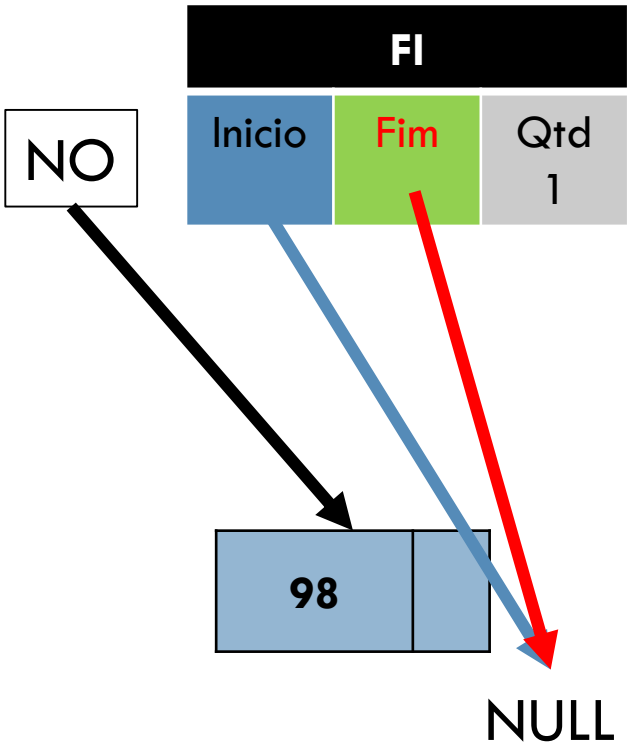

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



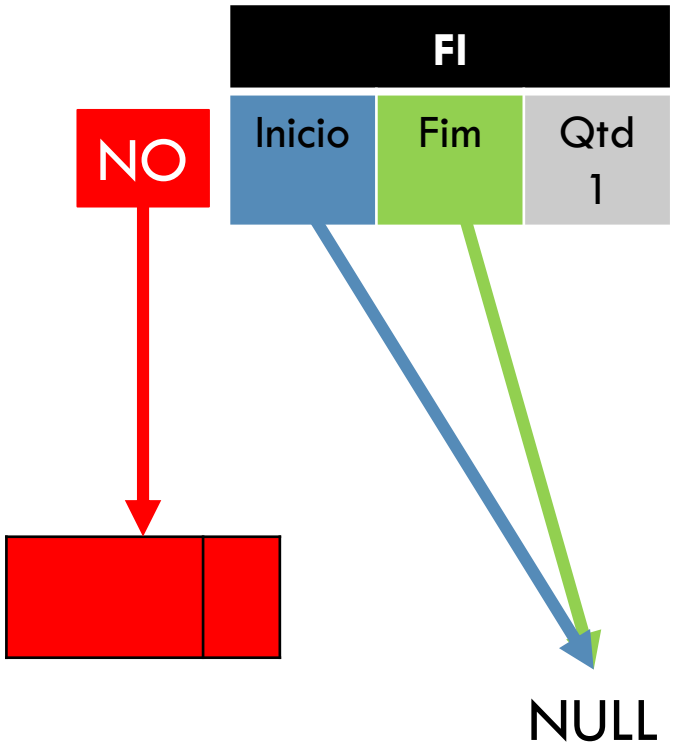

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção



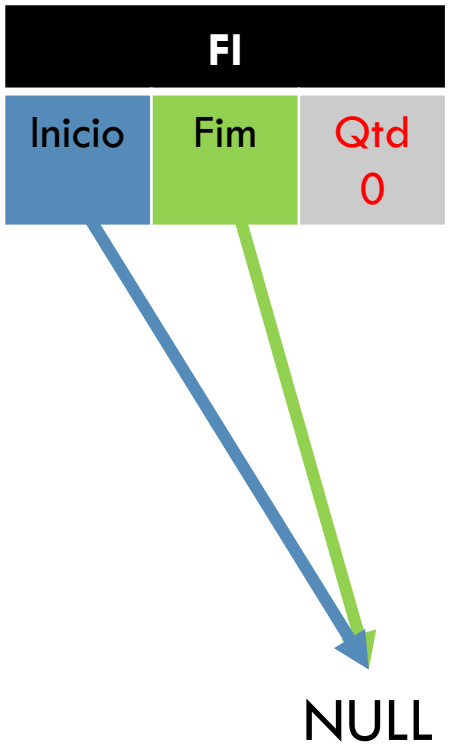

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
    fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}
```

Exemplo de Fila: Remoção



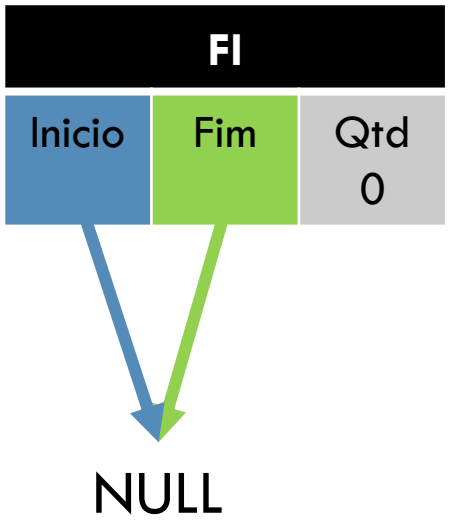

```
int remove_Fila(Fila* fi){
    if(fi == NULL)
        return 0;
    if(fi->inicio == NULL)//fila vazia
        return 0;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL)//fila ficou vazia
        fi->fim = NULL;
    free(no);
    fi->qtd--;
    return 1;
}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

Exemplo de Fila: Remoção




```
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




```
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;
}
```

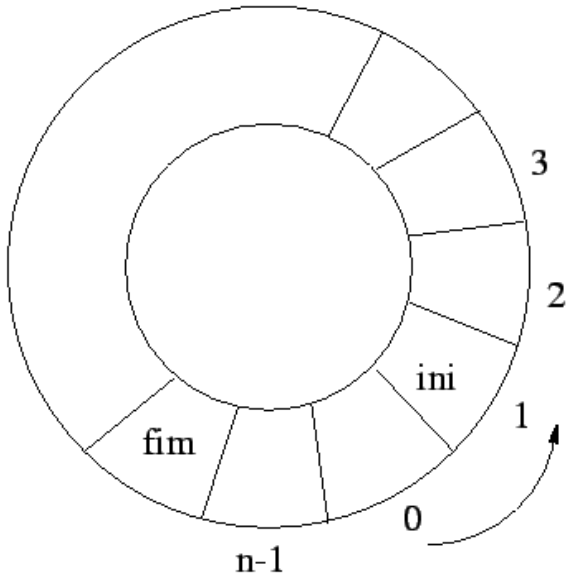
- **RETORNA OK;**
- **FINAL DA REMOÇÃO DO ELEMENTO.**

Fila Estática

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

Fila Estática

- 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.

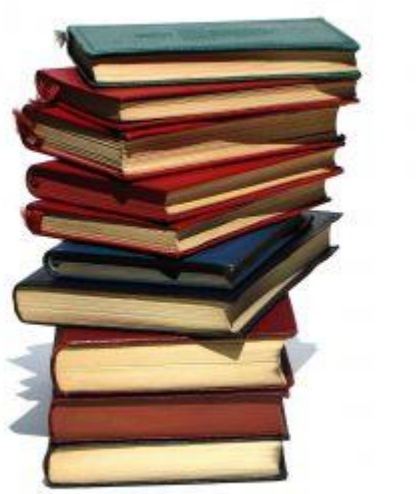


Fila: Exercícios

- 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?

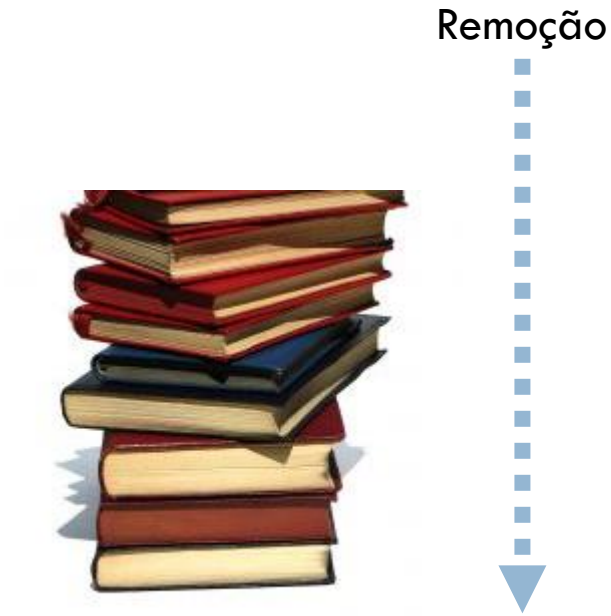
Pilha

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



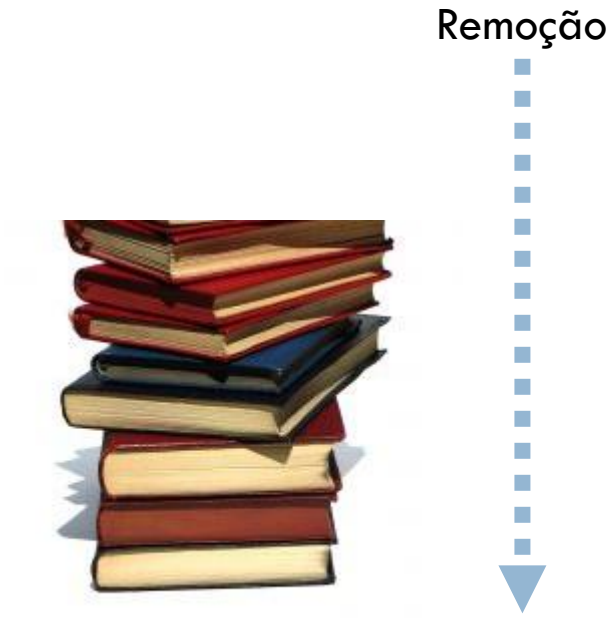
Pilha

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



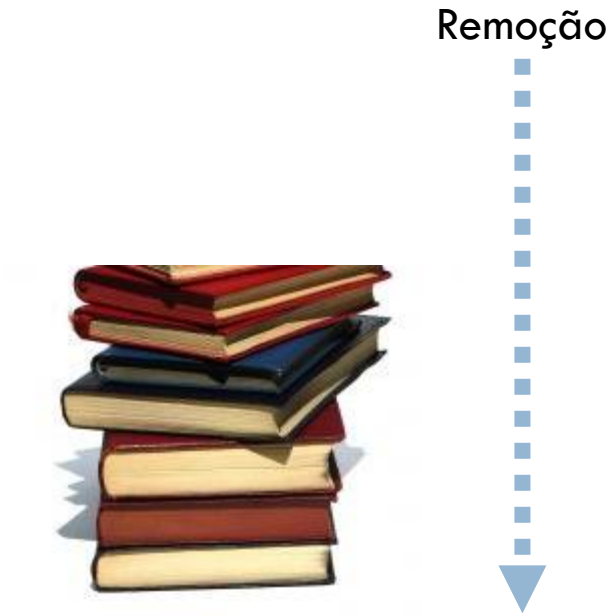
Pilha

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



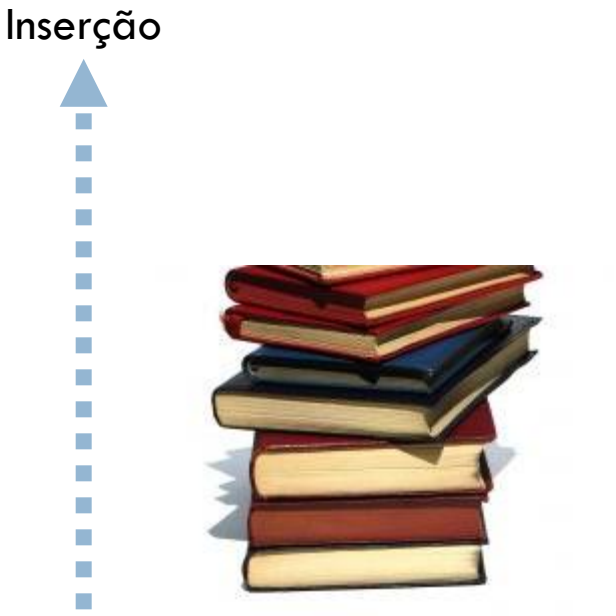
Pilha

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



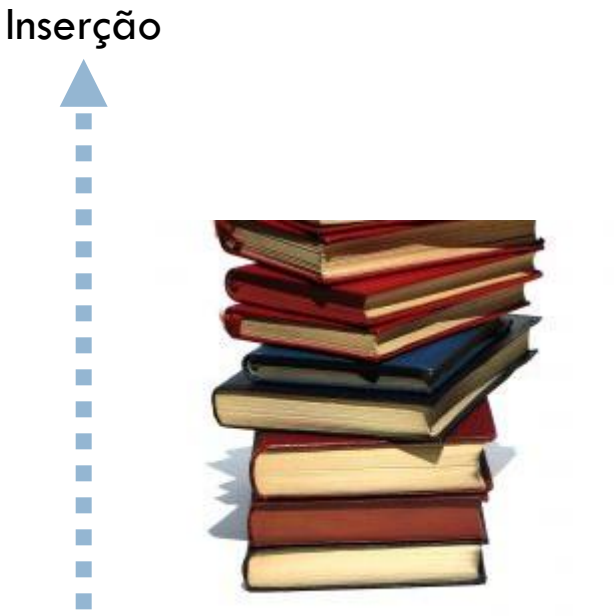
Pilha

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



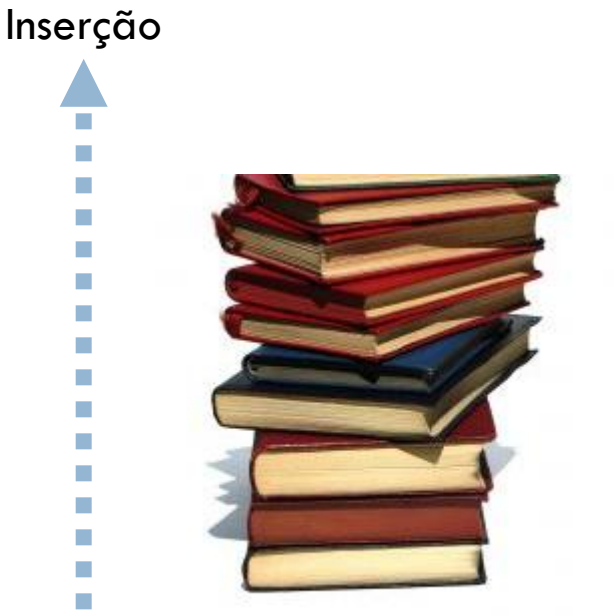
Pilha

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



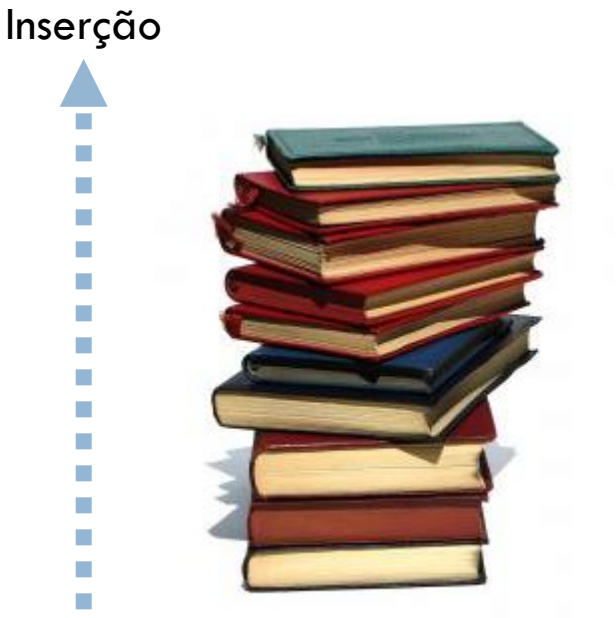
Pilha

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



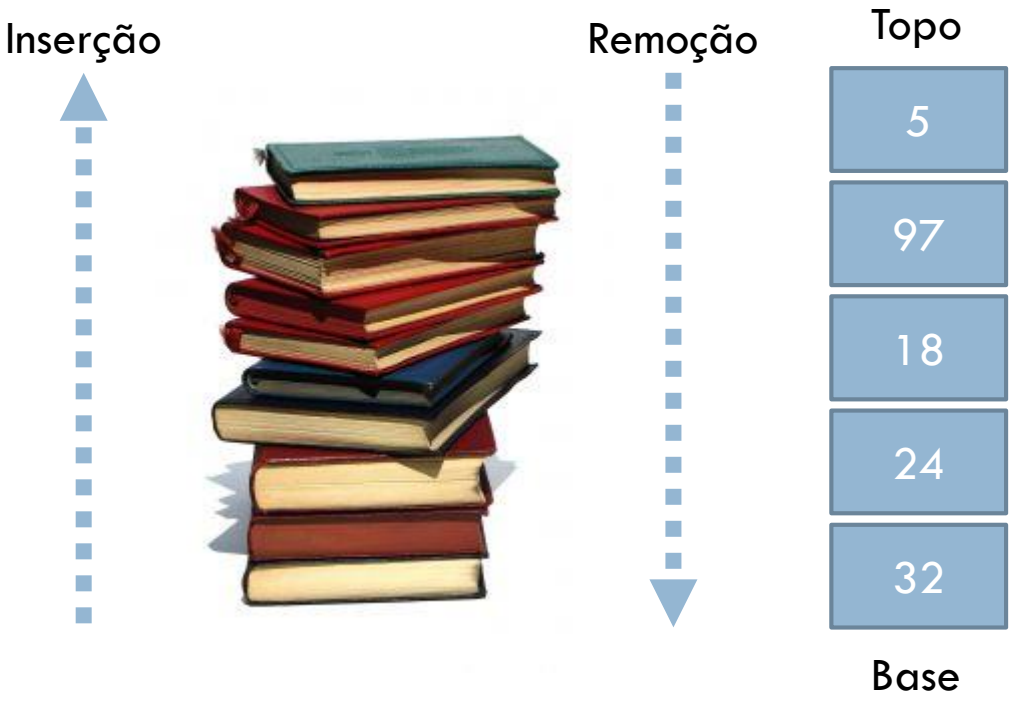
Pilha

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



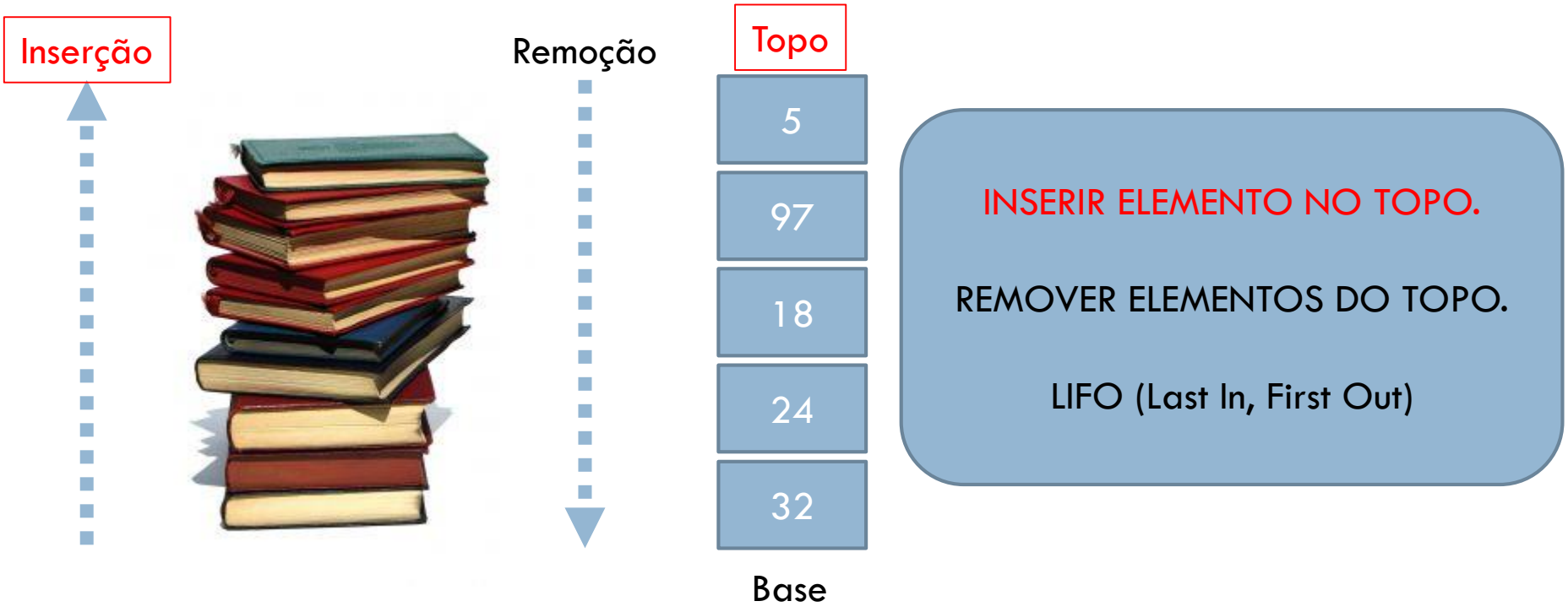
Pilha

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



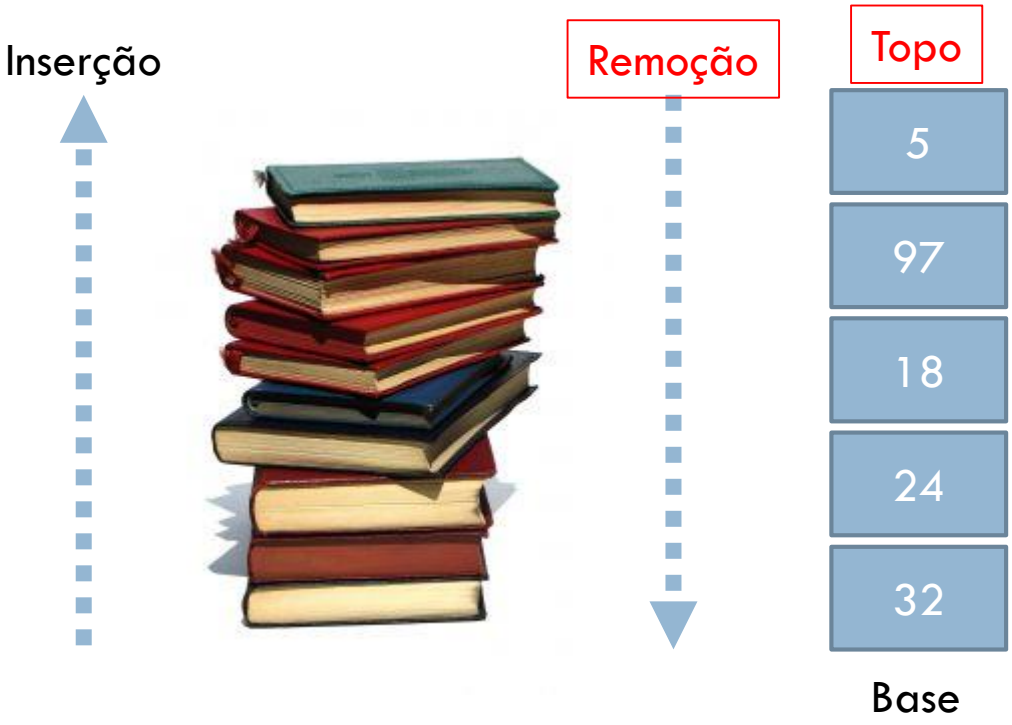
Pilha

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



Pilha

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



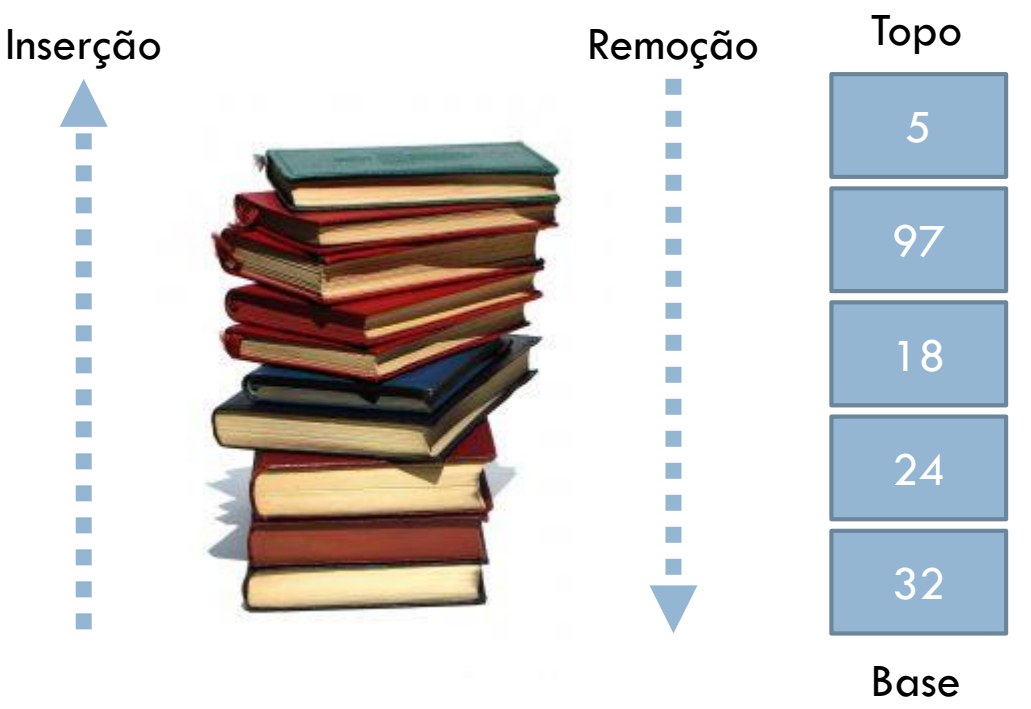
INSERIR ELEMENTO NO TOPO.

REMOVER ELEMENTOS DO TOPO.

LIFO (Last In, First Out)

Pilha

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



INSERIR ELEMENTO NO TOPO.

REMOVER ELEMENTOS DO TOPO.

LIFO (Last In, First Out)

Pilha

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

Pilha

- 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

- 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

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

Exemplo de Pilha: Definição de tipos


```
//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


```
//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);
```

IGUAL LISTA ENCADEADA DINÂMICA.

Exemplo de Pilha: Definição de tipos


```c
//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

IGUAL LISTA ENCADEADA DINÂMICA.

```c
//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: Criar/Liberar


```
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


```
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.

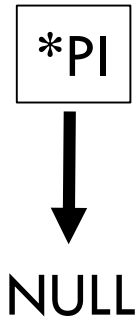
Exemplo de Pilha: Inserção


```
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

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;
}
```

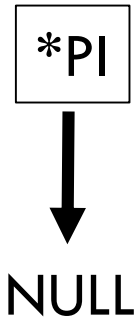


Exemplo de Pilha: Inserção

AL: 32

```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    *pi = no;
    return 1;
}
```

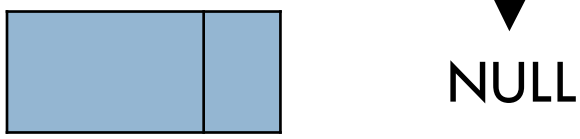
PILHA == NULL : NÃO ALOCADA.



Exemplo de Pilha: Inserção

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;
}
```



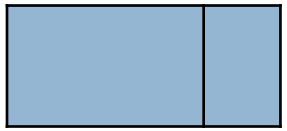
Exemplo de Pilha: Inserção

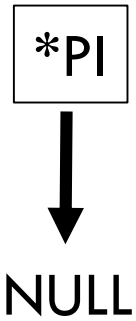
AL: 32

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

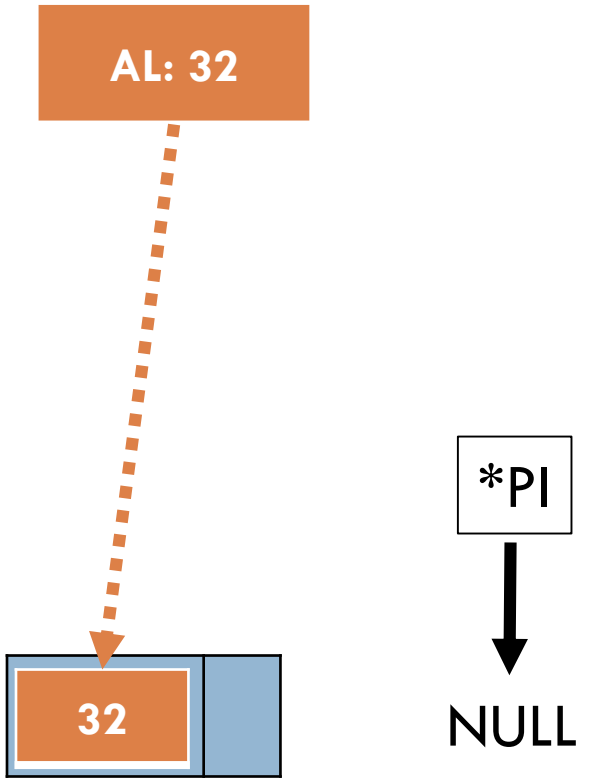
NO == NULL : NÃO ALOCADO.

```
}
```





Exemplo de Pilha: Inserção



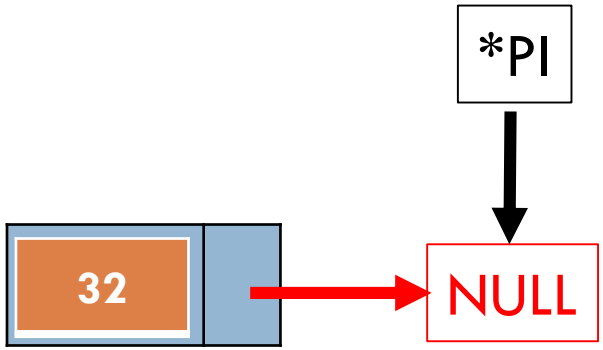

```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = (*pi);
    *pi = no;
    return 1;
}
```

Exemplo de Pilha: Inserção

AL: 32

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

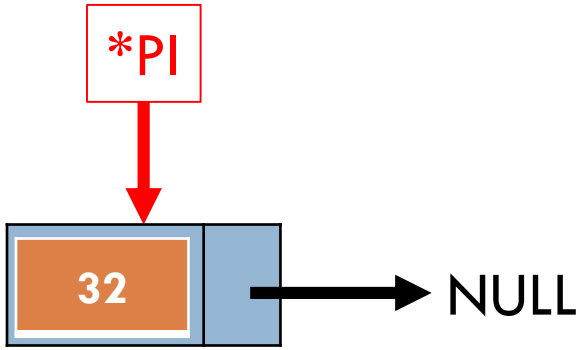
SEMPRE APONTA PARA O INICIO DA PILHA.



Exemplo de Pilha: Inserção

AL: 32

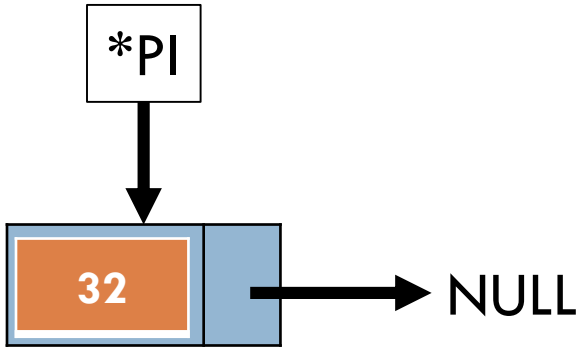
```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = (*pi);
    *pi = no;
    return 1;
}
```



Exemplo de Pilha: Inserção

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;
}
```

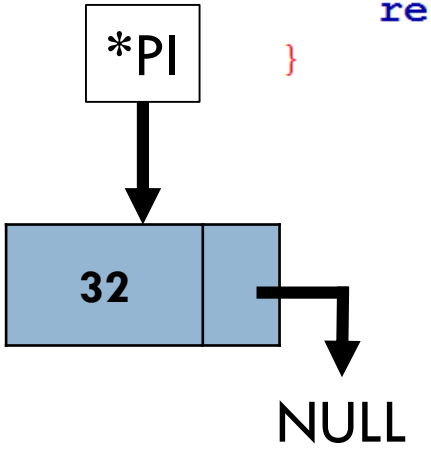


- **RETORNA OK;**
- **FINAL DA INSERÇÃO DO ELEMENTO.**

Exemplo de Pilha: Inserção

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;
}
```

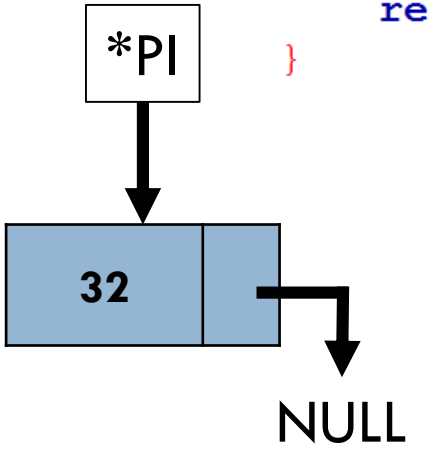


Exemplo de Pilha: Inserção

AL: 98

```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    *pi = no;
    return 1;
}
```

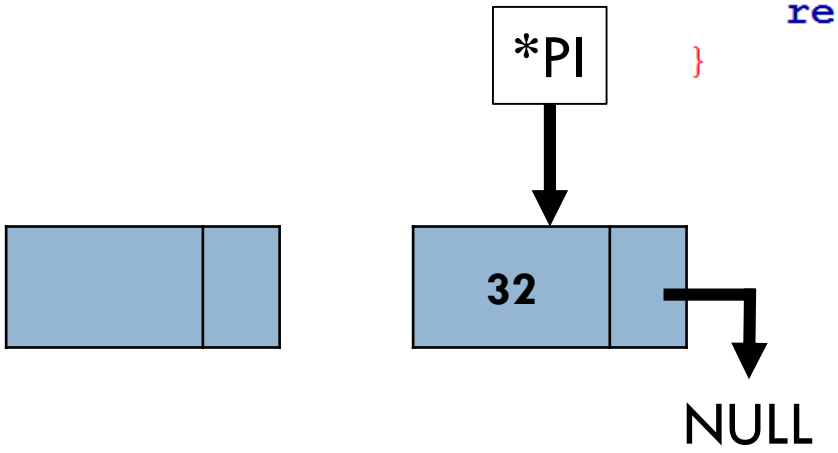
PILHA == NULL : NÃO ALOCADA.



Exemplo de Pilha: Inserção

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;
}
```



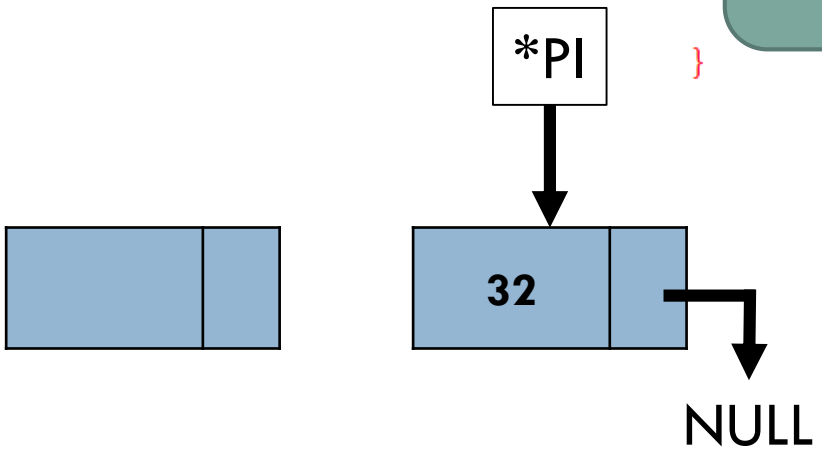
Exemplo de Pilha: Inserção

AL: 98

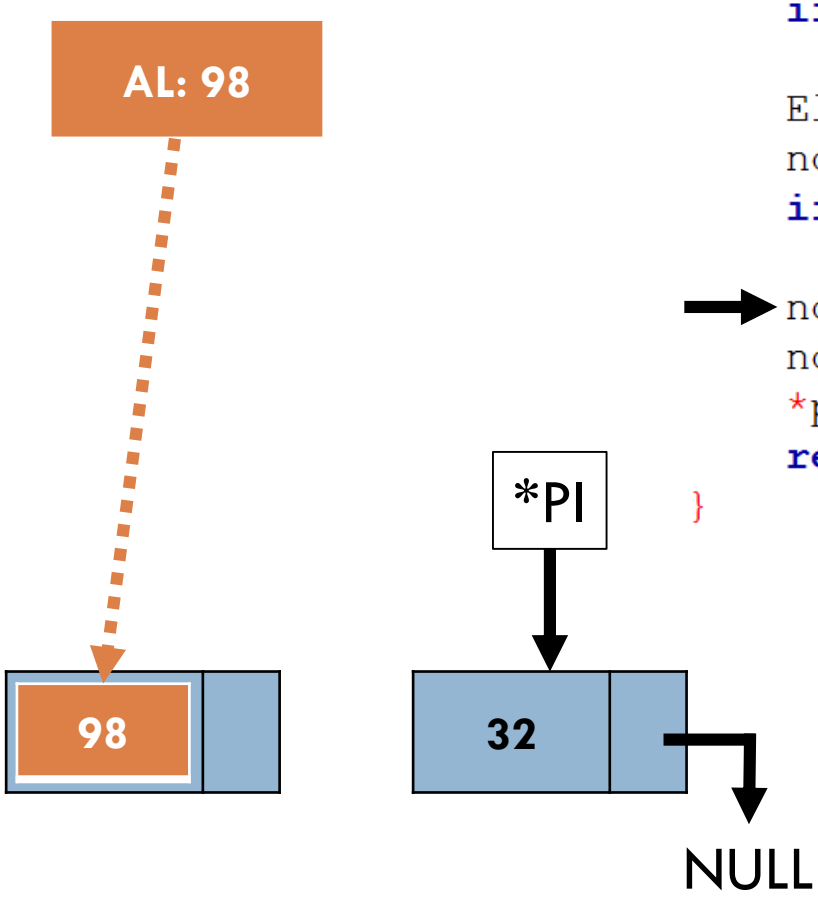
```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
```

NO == NULL : NÃO ALOCADO.

```
}
```



Exemplo de Pilha: Inserção



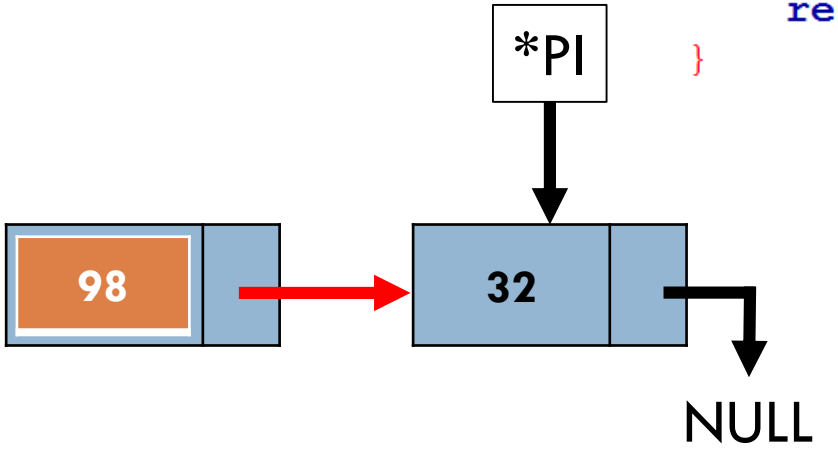

```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = (*pi);
    *pi = no;
    return 1;
}
```

Exemplo de Pilha: Inserção

AL: 98

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

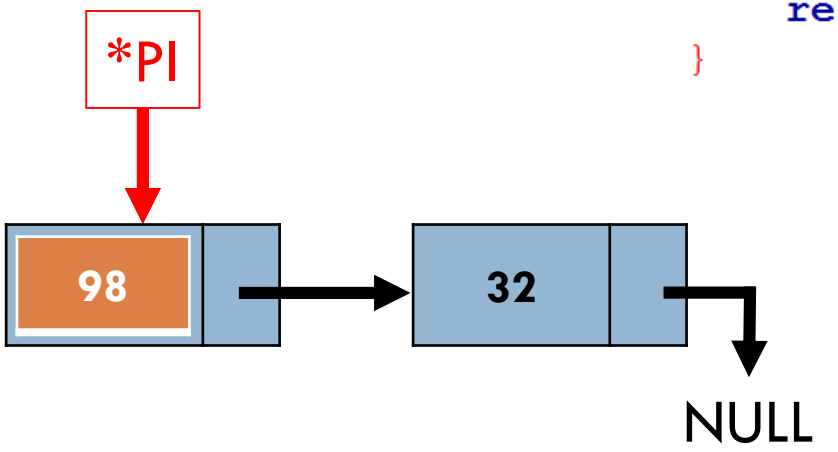
SEMPRE APONTA PARA O INICIO DA PILHA.



Exemplo de Pilha: Inserção

AL: 98

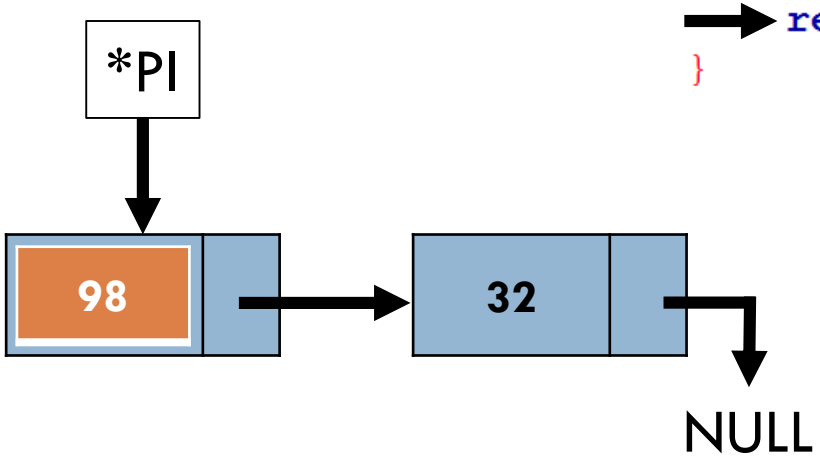
```
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = (*pi);
    *pi = no;
    return 1;
}
```



Exemplo de Pilha: Inserção

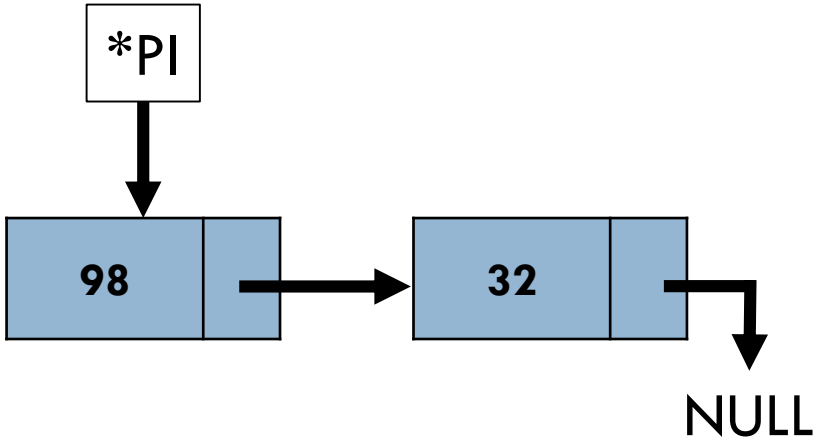
AL: 98

```
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;
}
```

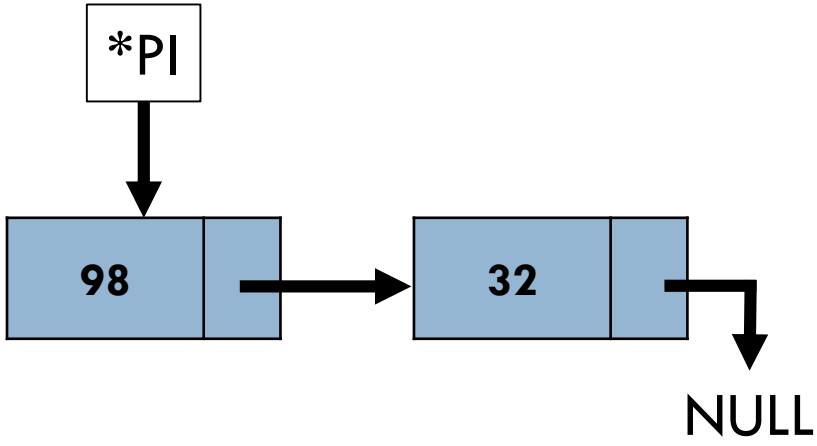


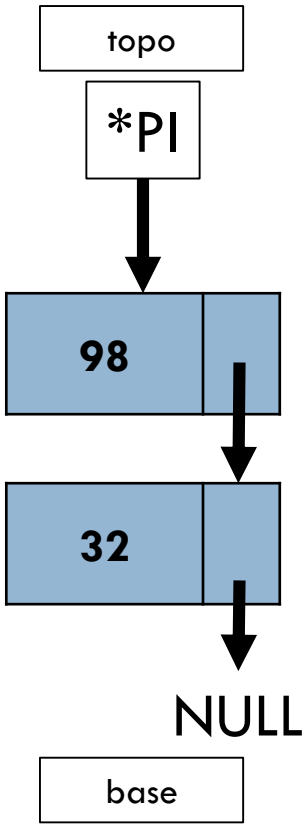
- **RETORNA OK;**
- **FINAL DA INSERÇÃO DO ELEMENTO.**

Exemplo de Pilha: Inserção



Exemplo de Pilha: Inserção



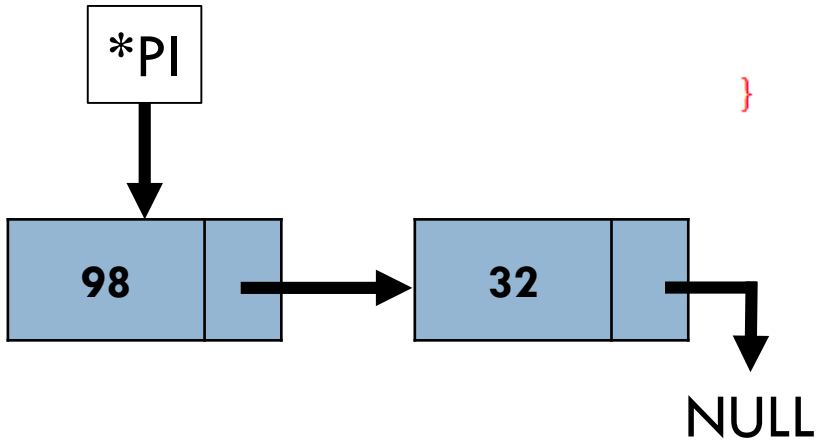


Exemplo de Pilha: Remoção


```
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


```
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;
}
```

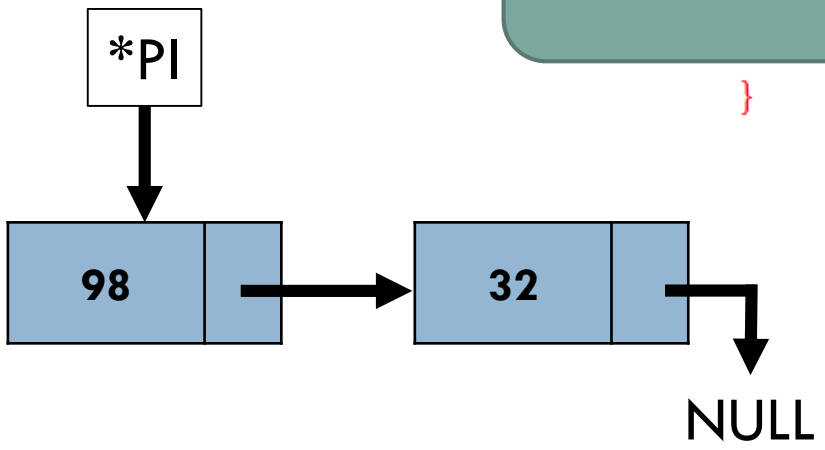


Exemplo de Pilha: Remoção


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

PILHA == NULL : NÃO ALOCADA.

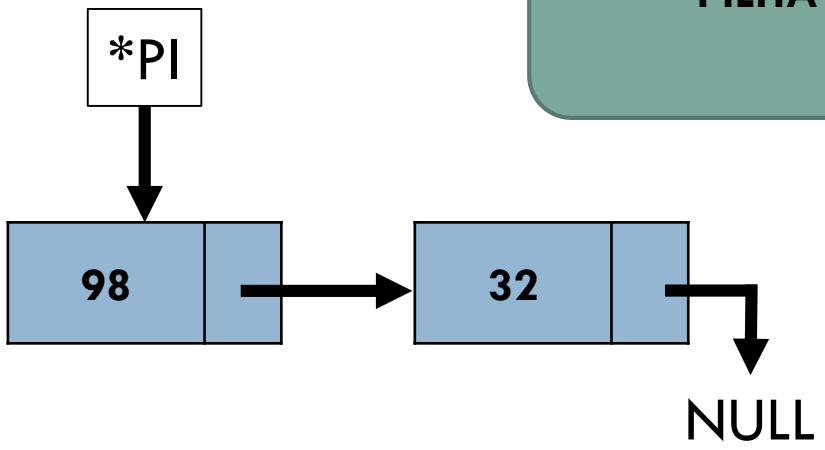
```
}
```



Exemplo de Pilha: Remoção

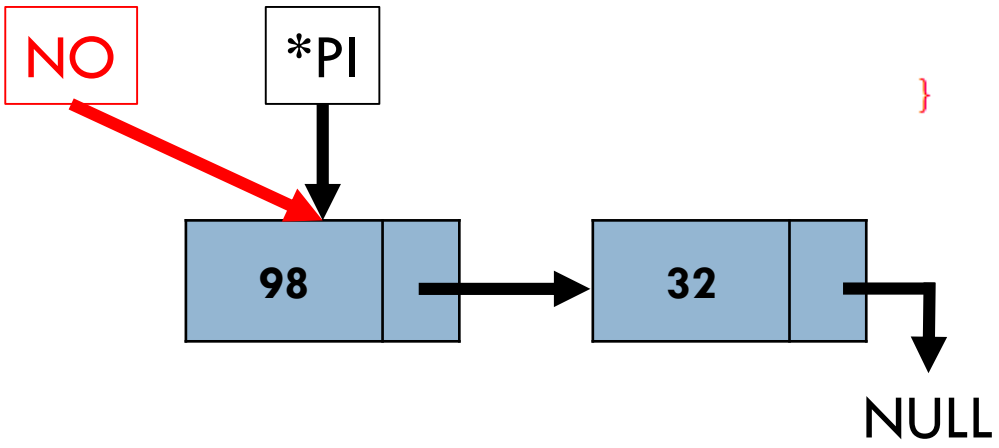

```
int remove_Pilha(Pilha* pi){
    if(pi == NULL)
        return 0;
    if((*pi) == NULL)
        return 0;
```

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



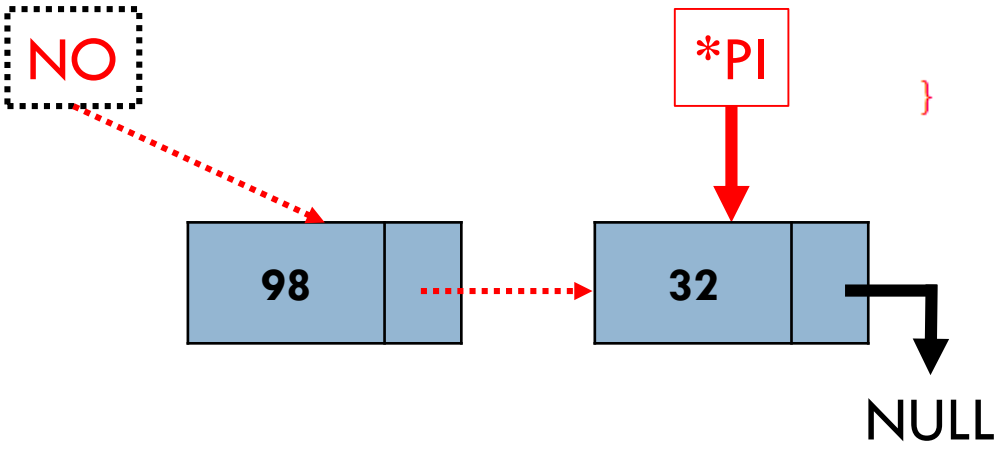
Exemplo de Pilha: Remoção


```
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;
}
```



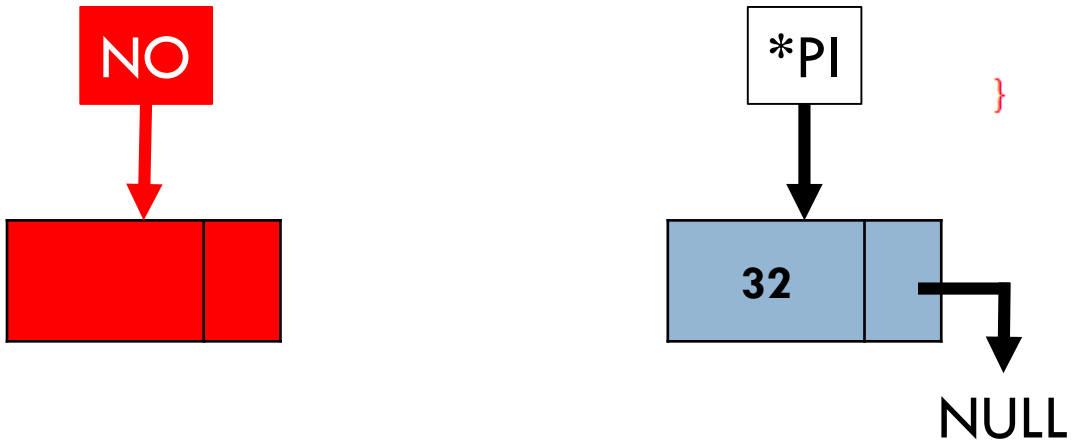
Exemplo de Pilha: Remoção


```
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;
}
```



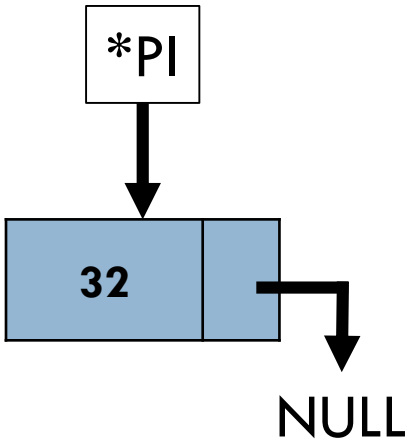
Exemplo de Pilha: Remoção


```
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;
}
```



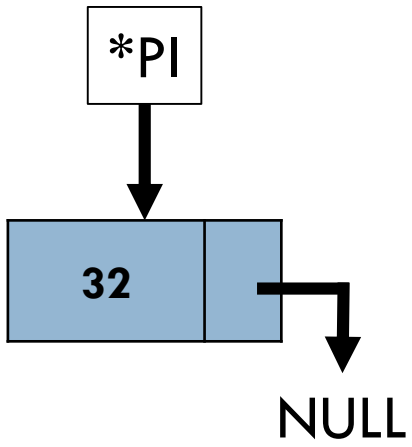
Exemplo de Pilha: Remoção

- **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



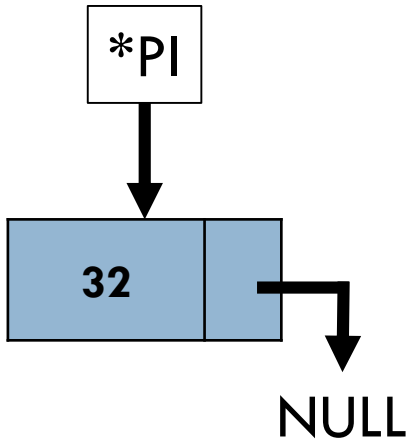

```
int remove_Pilha(Pilha* pi){
    if(pi == NULL)
        return 0;
    if((*pi) == NULL)
        return 0;
    Elem *no = *pi;
    *pi = no->prox;
    free(no);
    return 1;
}
```

Exemplo de Pilha: Remoção


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

PILHA == NULL : NÃO ALOCADA.

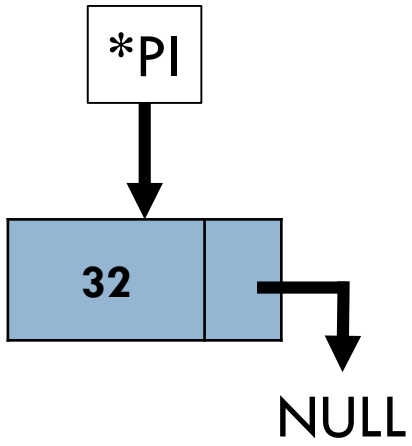
```
}
```



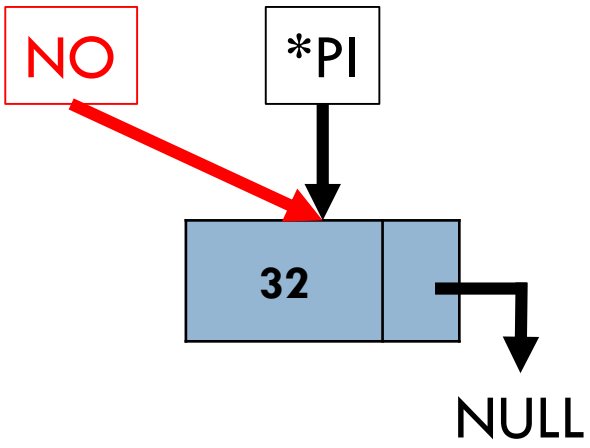
Exemplo de Pilha: Remoção


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

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

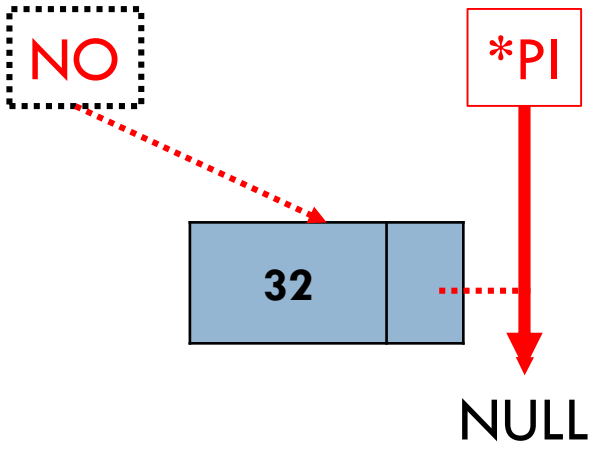


Exemplo de Pilha: Remoção



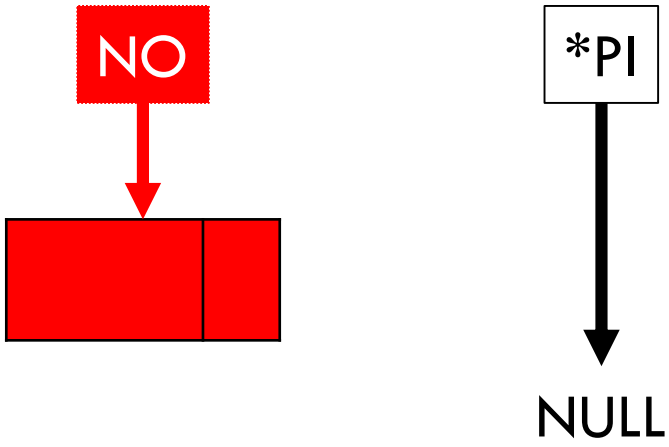

```
int remove_Pilha(Pilha* pi){
    if(pi == NULL)
        return 0;
    if((*pi) == NULL)
        return 0;
    Elem *no = *pi;
    *pi = no->prox;
    free(no);
    return 1;
}
```

Exemplo de Pilha: Remoção




```
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;
}
```

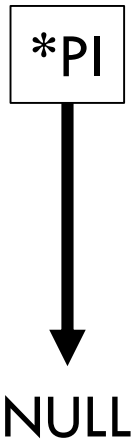
Exemplo de Pilha: Remoção




```
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

- **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

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

Pilha: Exercícios

- 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.**

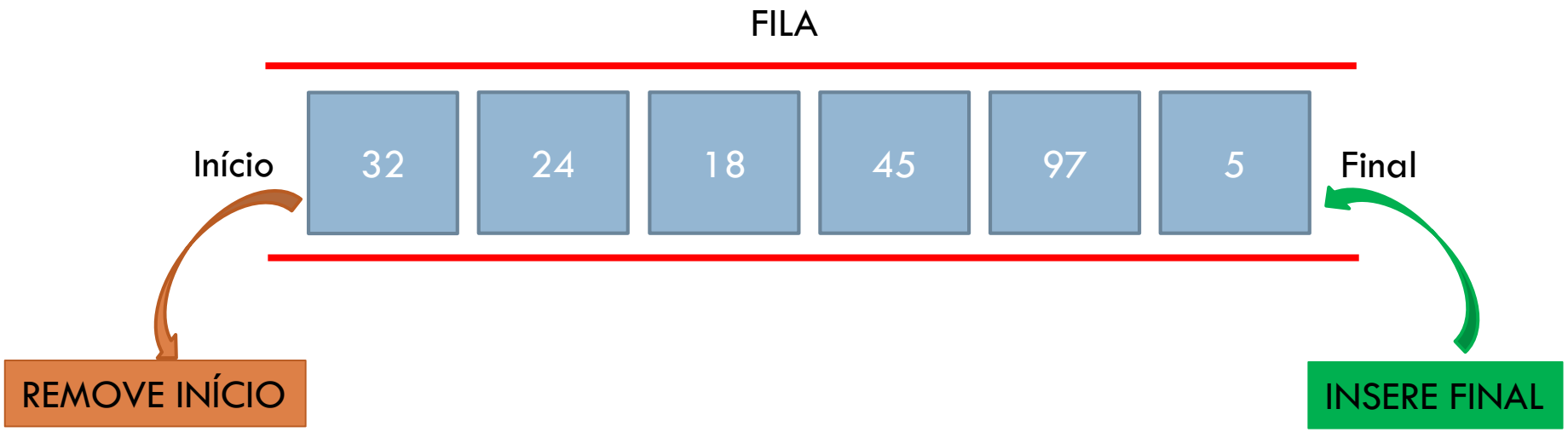
Deques

- 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.

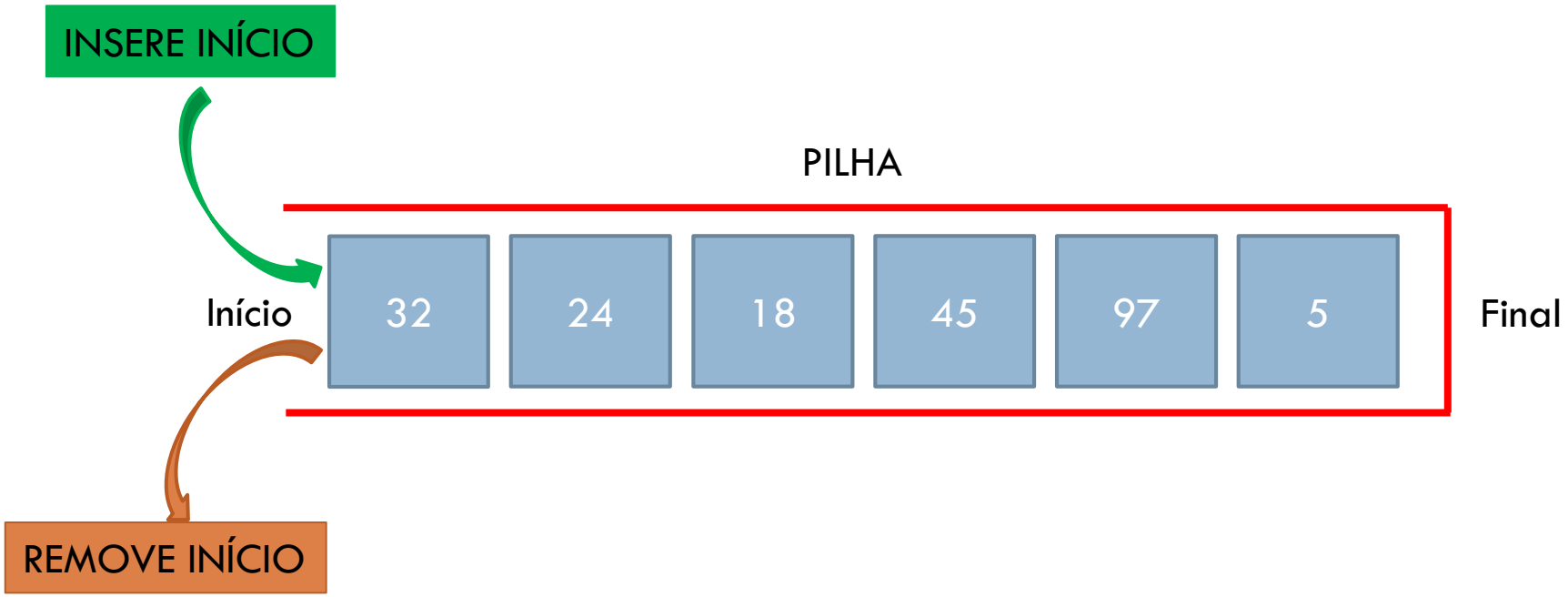
Deques

- FILA:



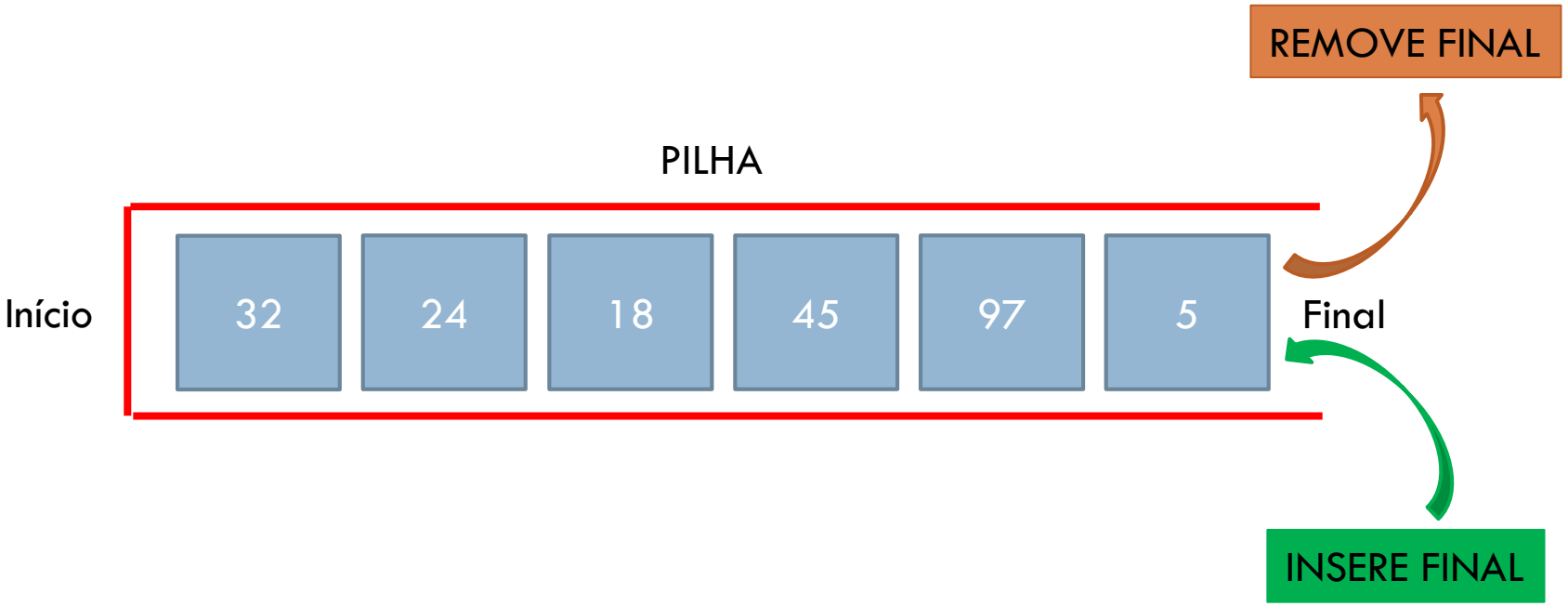
Deques

- PILHA.



Deques

- PILHA.



Deques

- Inserção e remoção de ambas as extremidades.