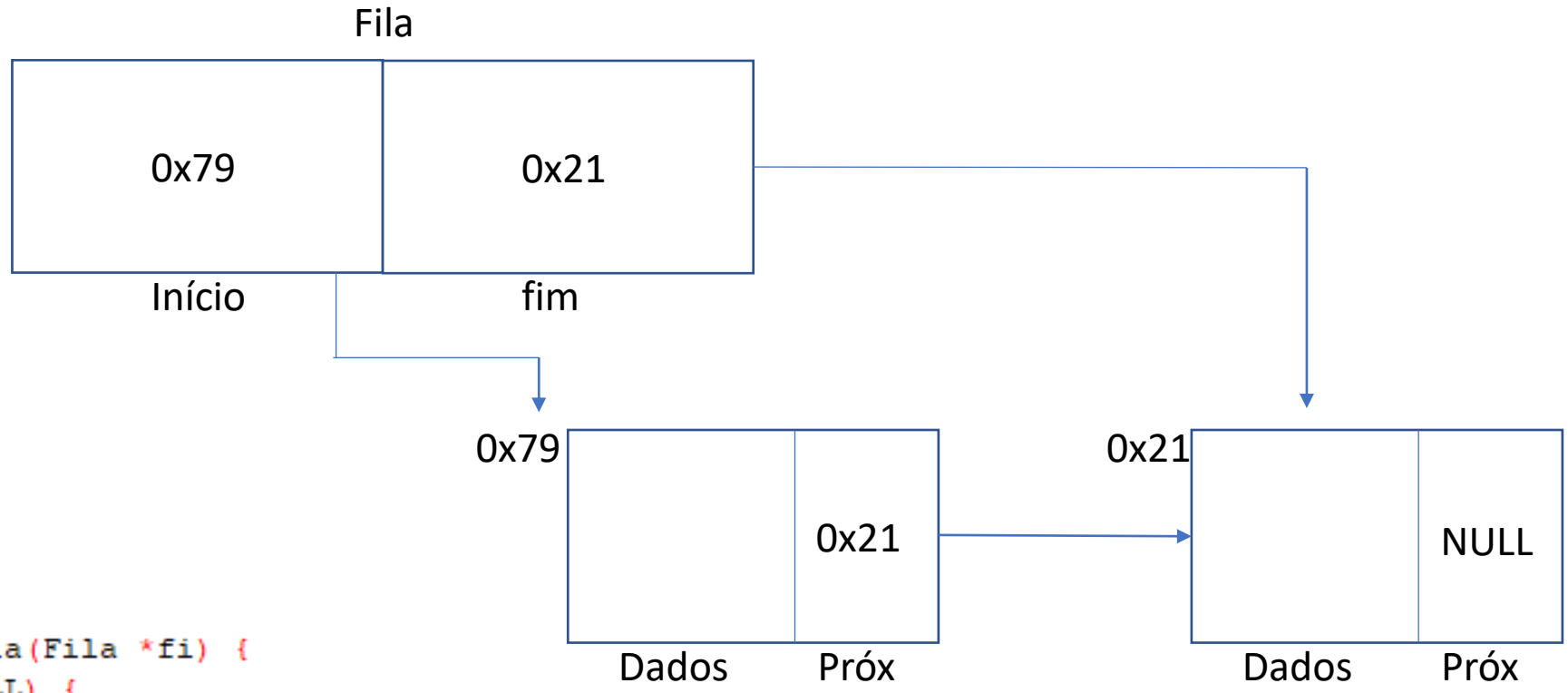
Fila
operações passo a passo

Destruindo a fila

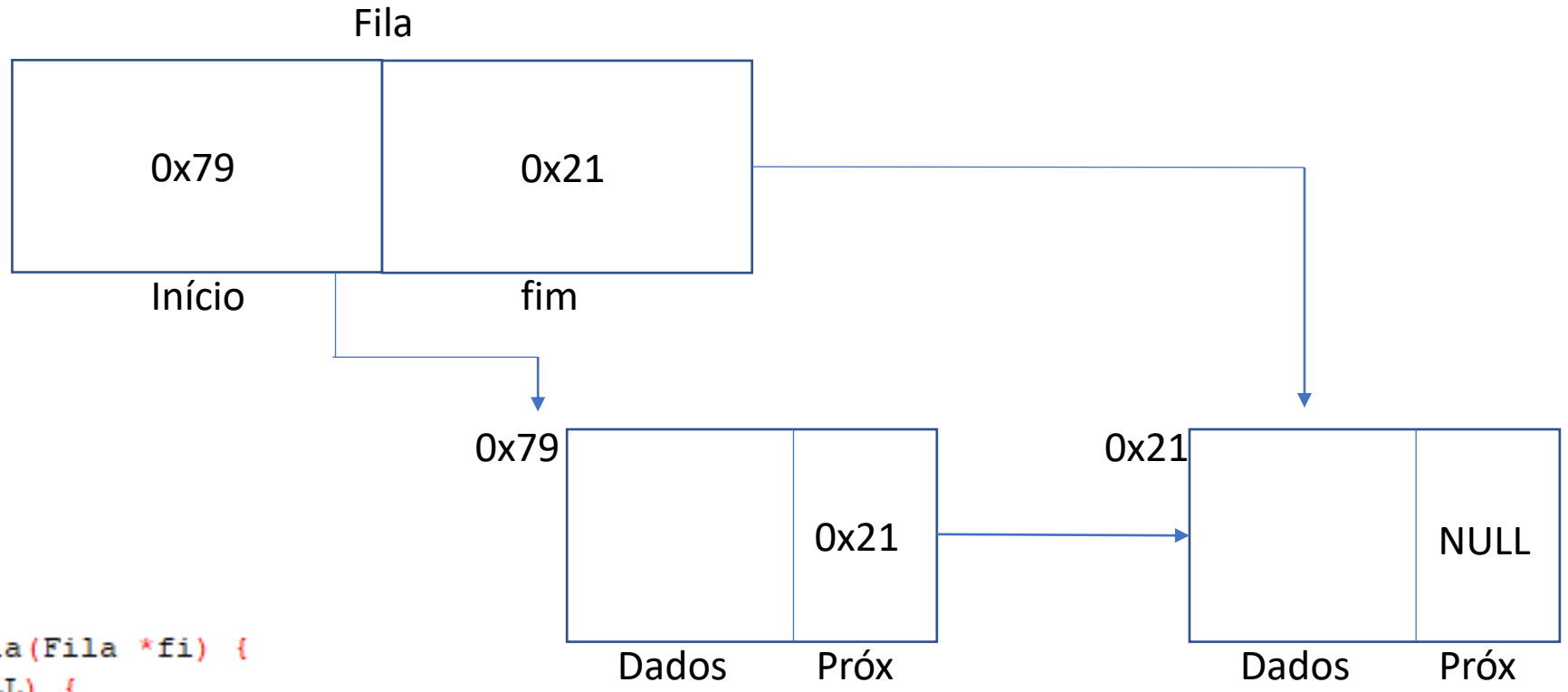
estado inicial



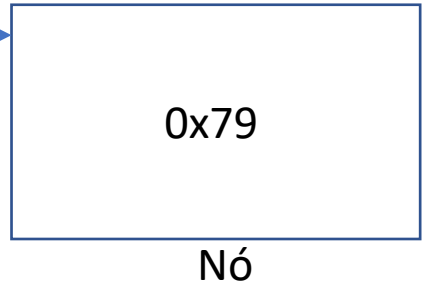

```
void destroi_fila(Fila *fi) {
    if(fi != NULL) {
        Elem *no;
        while(fi->inicio != NULL) {
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```

Destruindo a fila

início é nulo? Não, então, nó recebe o início da fila

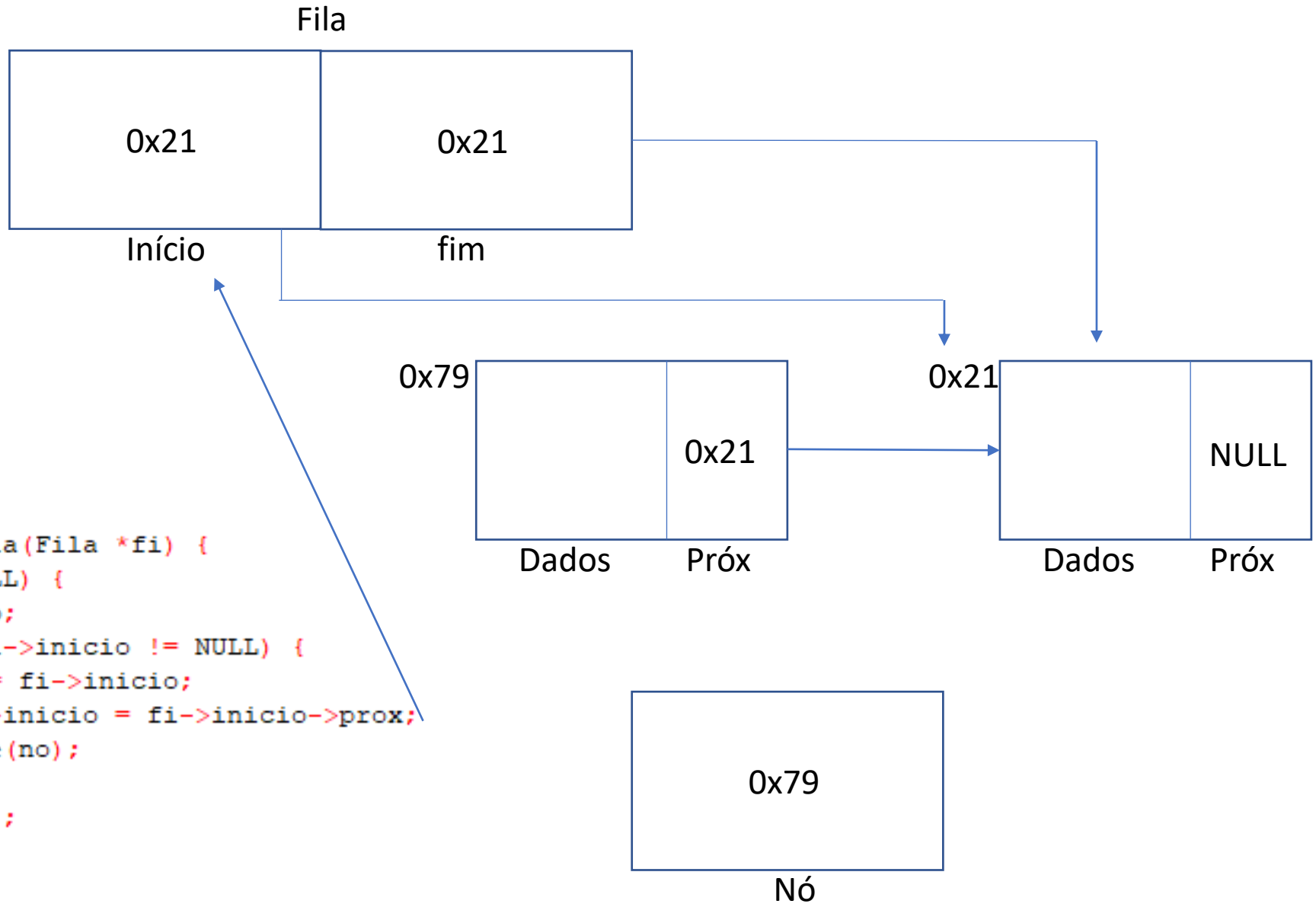



```
void destroi_fila(Fila *fi) {
    if(fi != NULL) {
        Elem *no;
        while(fi->inicio != NULL) {
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```



Destruindo a fila

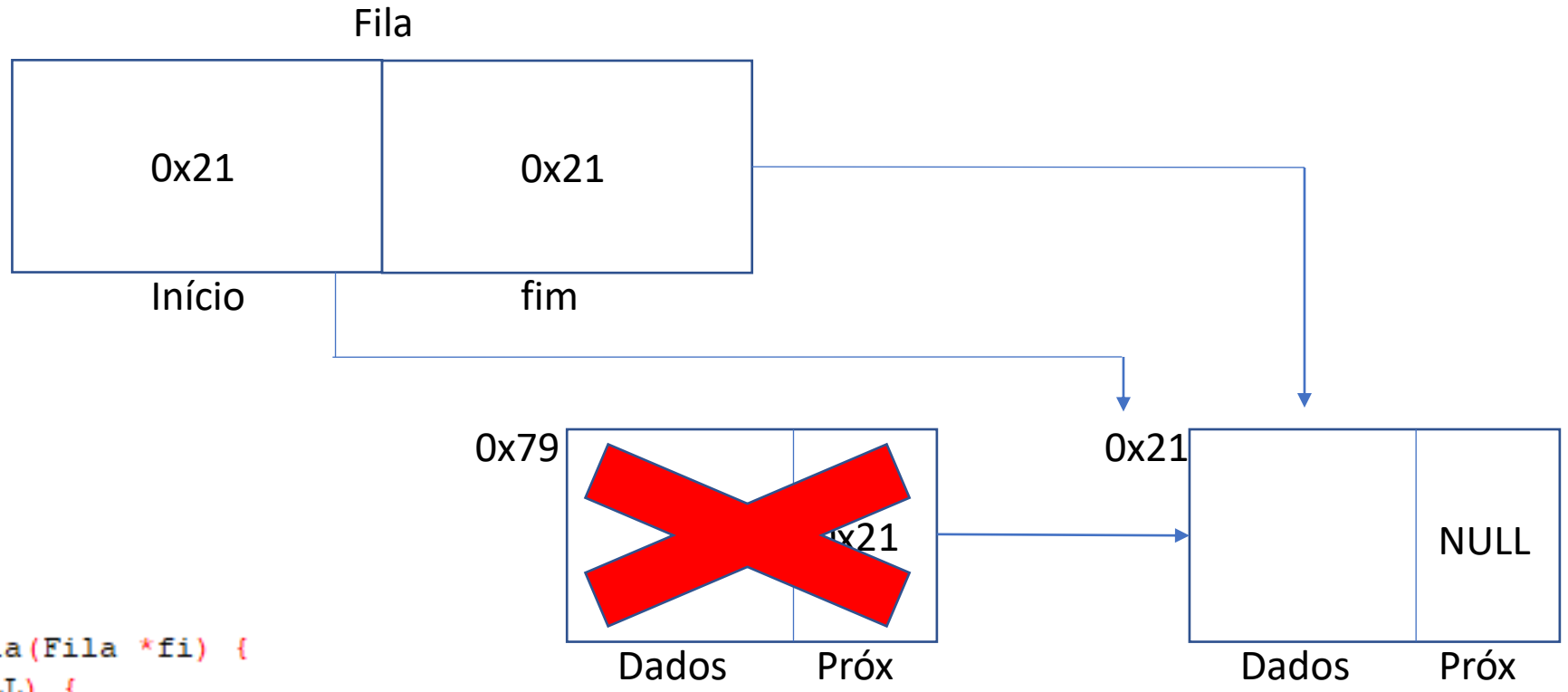
início da fila recebe o endereço do próximo elemento (0x21)



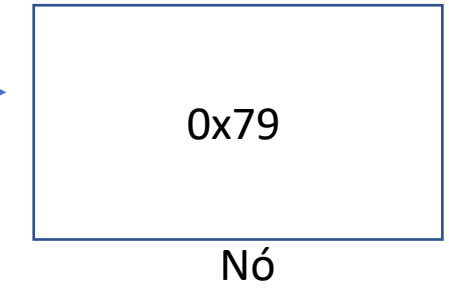

```
void destroi_fila(Fila *fi) {
    if(fi != NULL) {
        Elem *no;
        while(fi->inicio != NULL) {
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```

Destruindo a fila

Agora basta liberar o elemento onde o nó aponta (0x79)

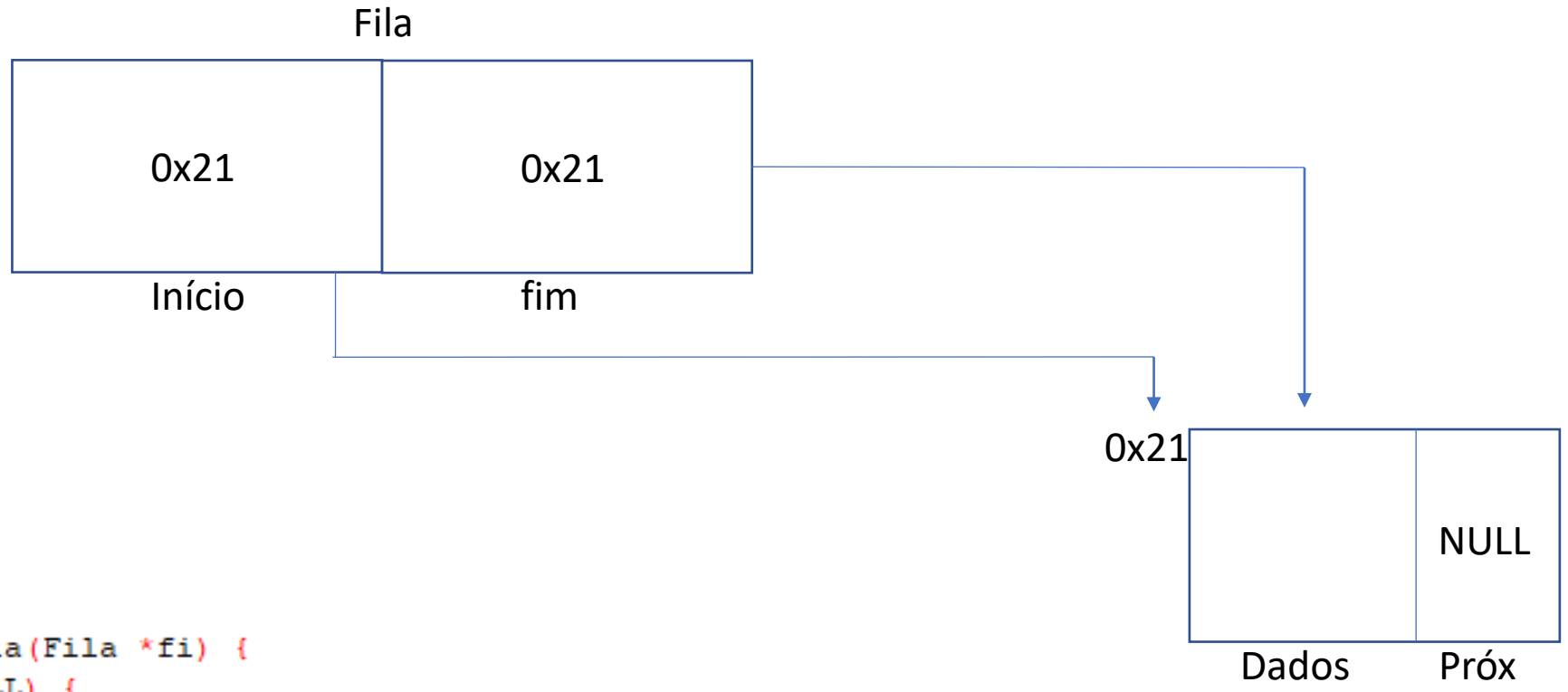



```
void destroi_fila(Fila *fi) {
    if(fi != NULL) {
        Elem *no;
        while(fi->inicio != NULL) {
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```

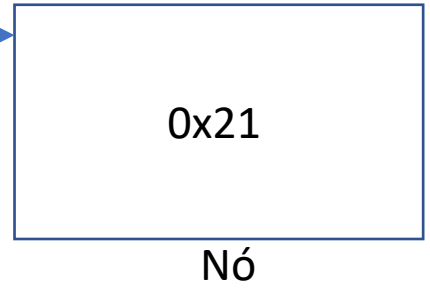


Destruindo a fila

Início é nulo? Não, então repetimos o processo

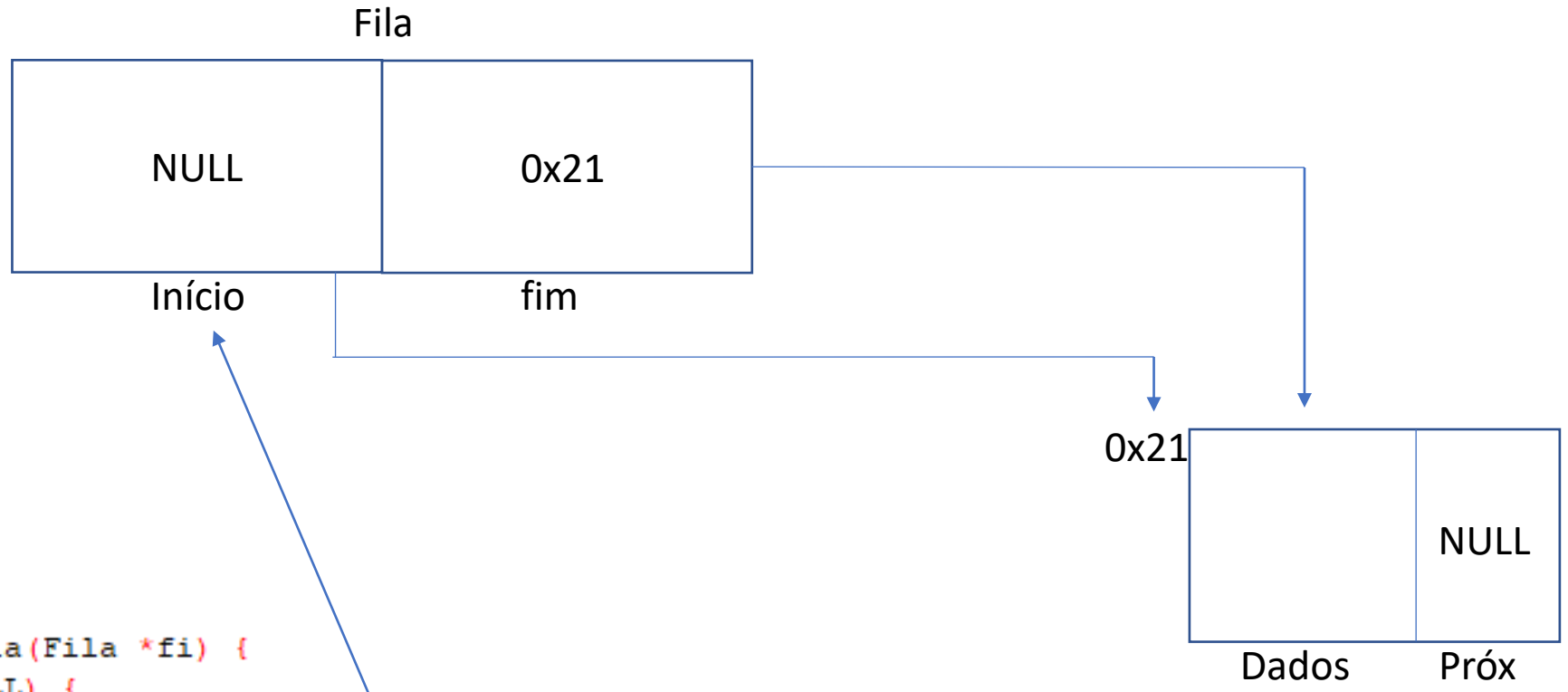



```
void destroi_fila(Fila *fi) {
    if(fi != NULL) {
        Elem *no;
        while(fi->inicio != NULL) {
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```



Destruindo a fila

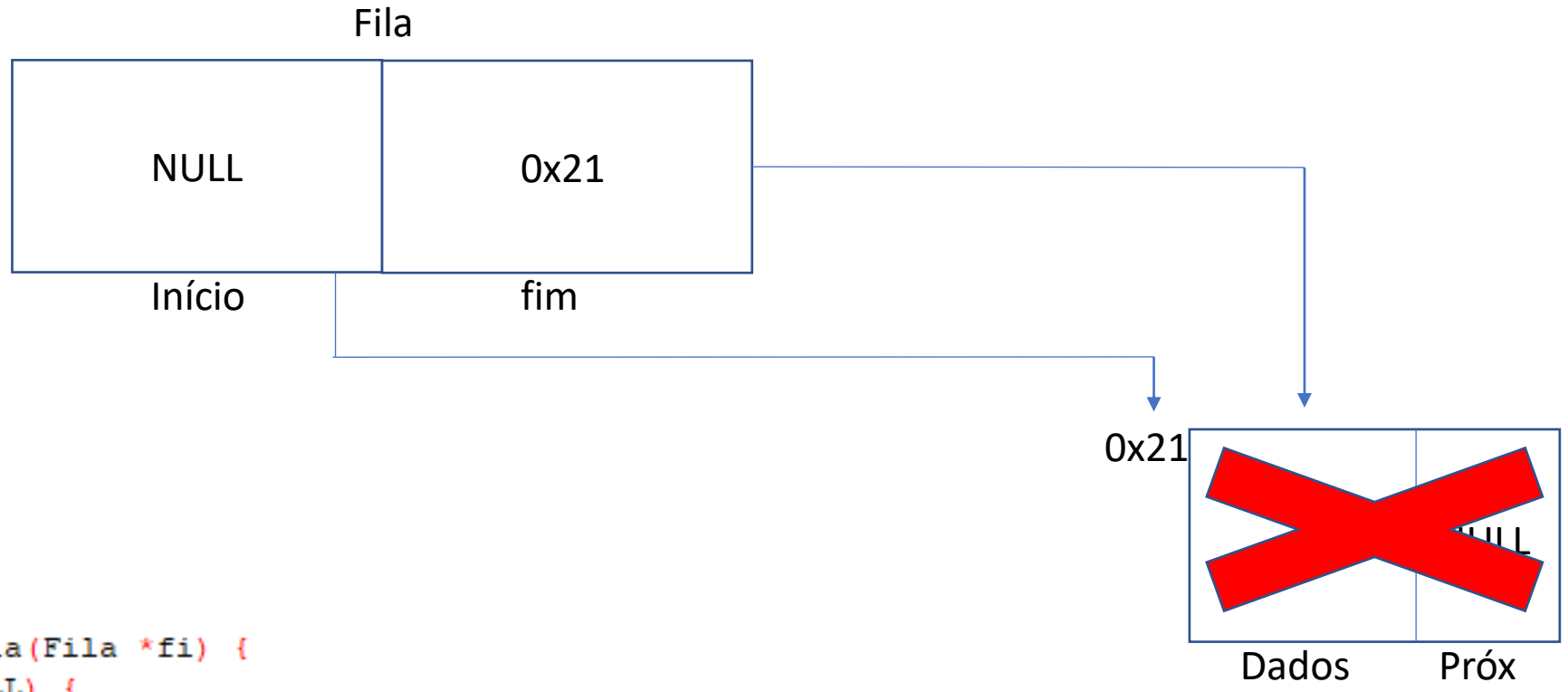
início da fila recebe o endereço do próximo elemento (NULL)



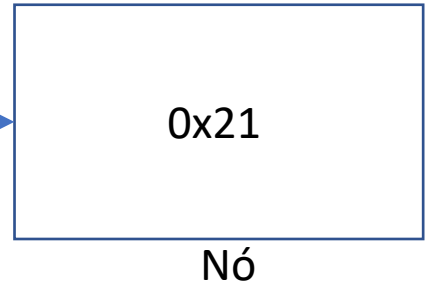

```
void destroi_fila(Fila *fi) {
    if(fi != NULL) {
        Elem *no;
        while(fi->inicio != NULL) {
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```

Destruindo a fila

Agora basta liberar o elemento onde o nó aponta (0x21)

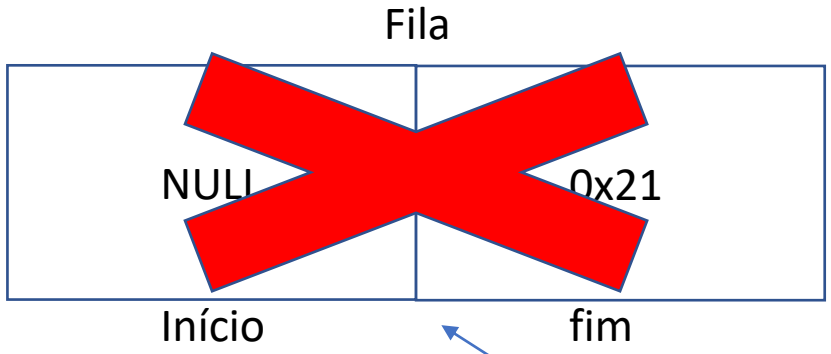



```
void destroi_fila(Fila *fi) {
    if(fi != NULL) {
        Elem *no;
        while(fi->inicio != NULL) {
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}
```



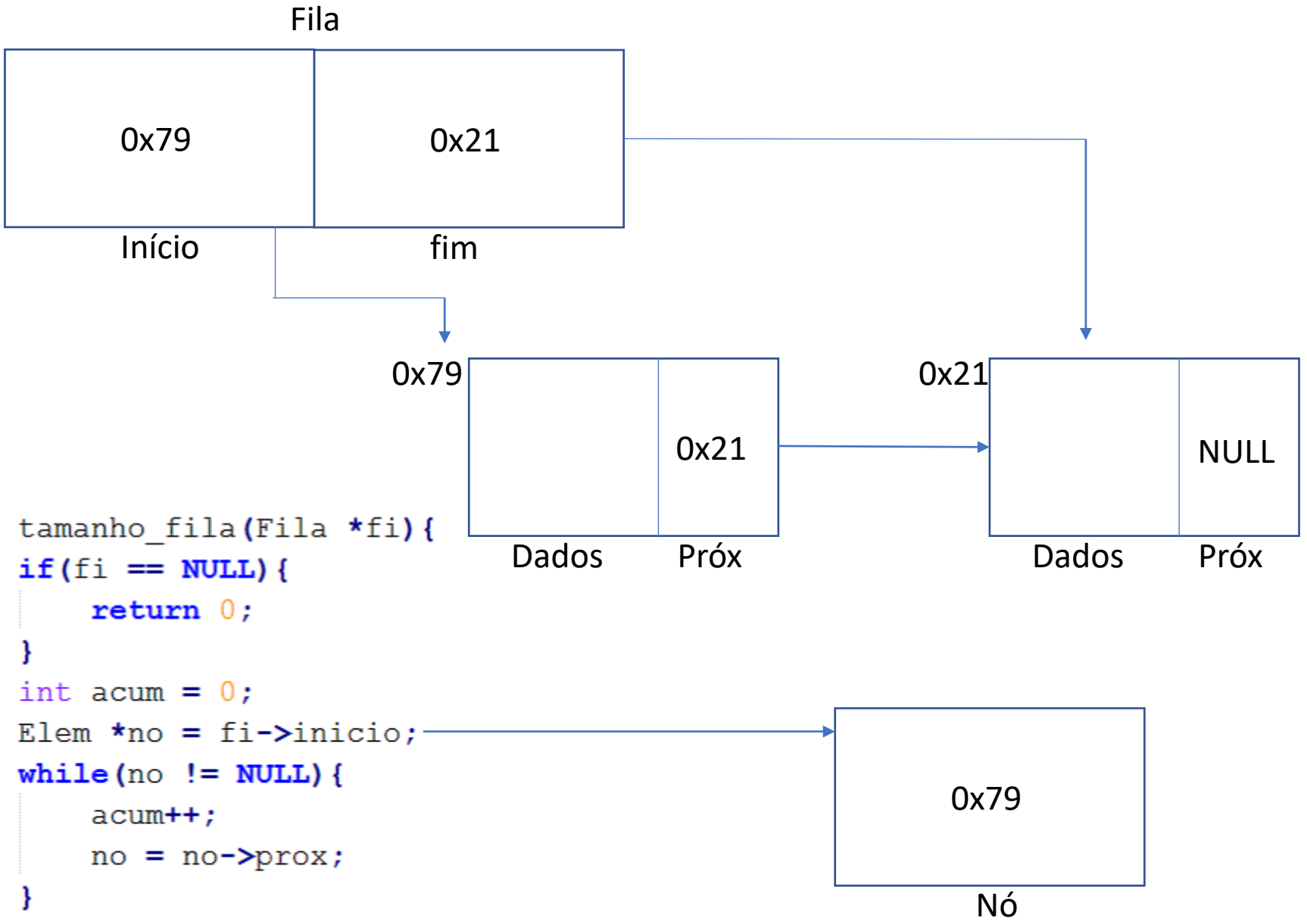
Destruindo a fila

Agora o início aponta pra nulo, basta destruir a fila




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

Calculando tamanho da fila




```c
int tamanho_fila(Fila *fi){
    if(fi == NULL){
        return 0;
    }
    int acum = 0;
    Elem *no = fi->inicio;
    while(no != NULL){
        acum++;
        no = no->prox;
    }
    return acum;
}
```

1ª iteração
Nó é diferente de nulo?

Nó vale 0x79, logo é diferente de nulo.

acum++ (acumulador era 0, passa a valer 1)

Nó = nó->prox (0x21)

2º iteração
Nó é diferente de nulo?

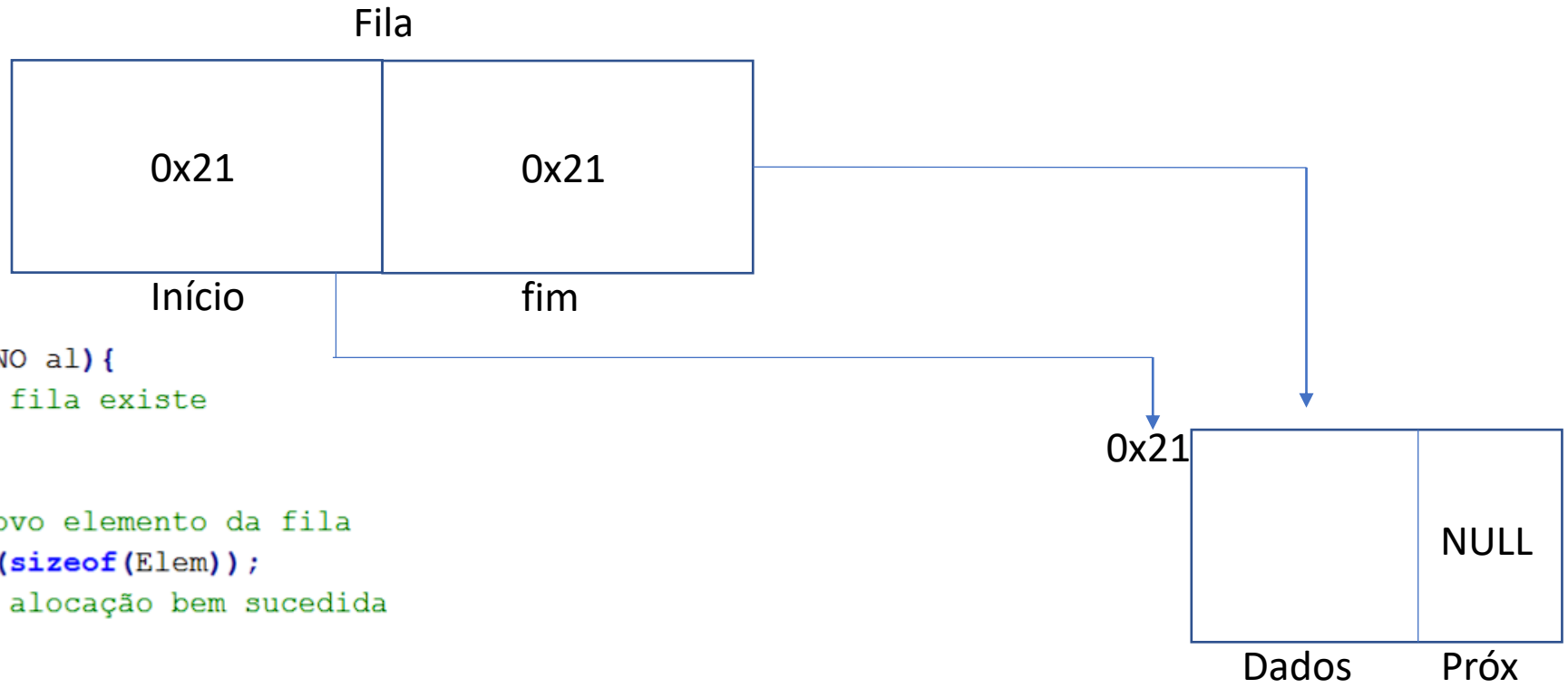
Nó vale 0x21, logo é diferente de nulo.

acum++ (acumulador era 1, passa a valer 2)

Nó = nó->prox (NULL)

3ª iteração
Nó = NULL, logo sai do loop e retorna acum (2)

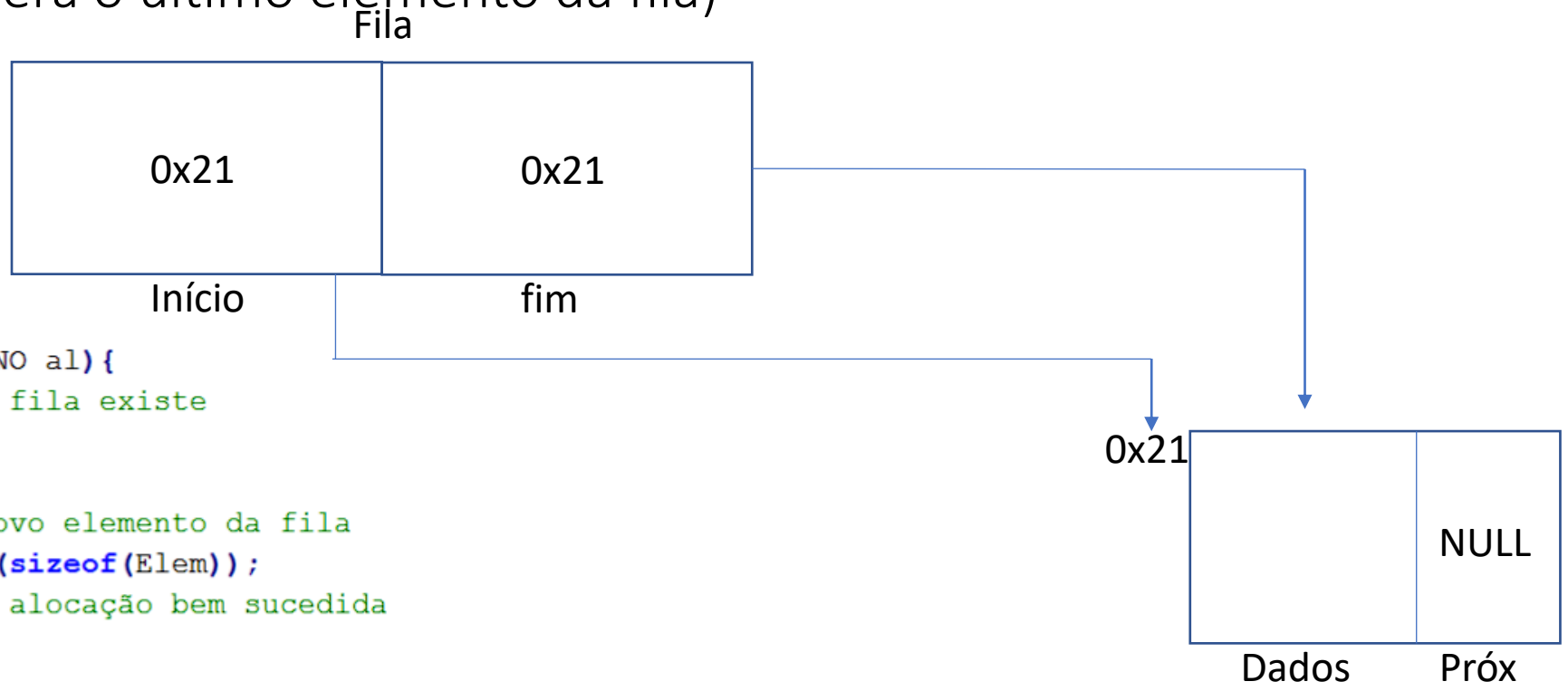
Inserção na fila
estado inicial




```
int insere_fila(Fila *fi, ALUNO al){
    if(fi == NULL){//Testa se fila existe
        return 0;
    }
    //aloca memória para um novo elemento da fila
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL){//testa se alocação bem sucedida
        return 0;
    }
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL){//fila vazia
        fi->inicio = no;
    }else{ //insere no final da fila
        fi->fim->prox = no;
    }
    fi->fim = no; //no passa a ser novo final da fila
    return 1;
}
```

Inserção na fila

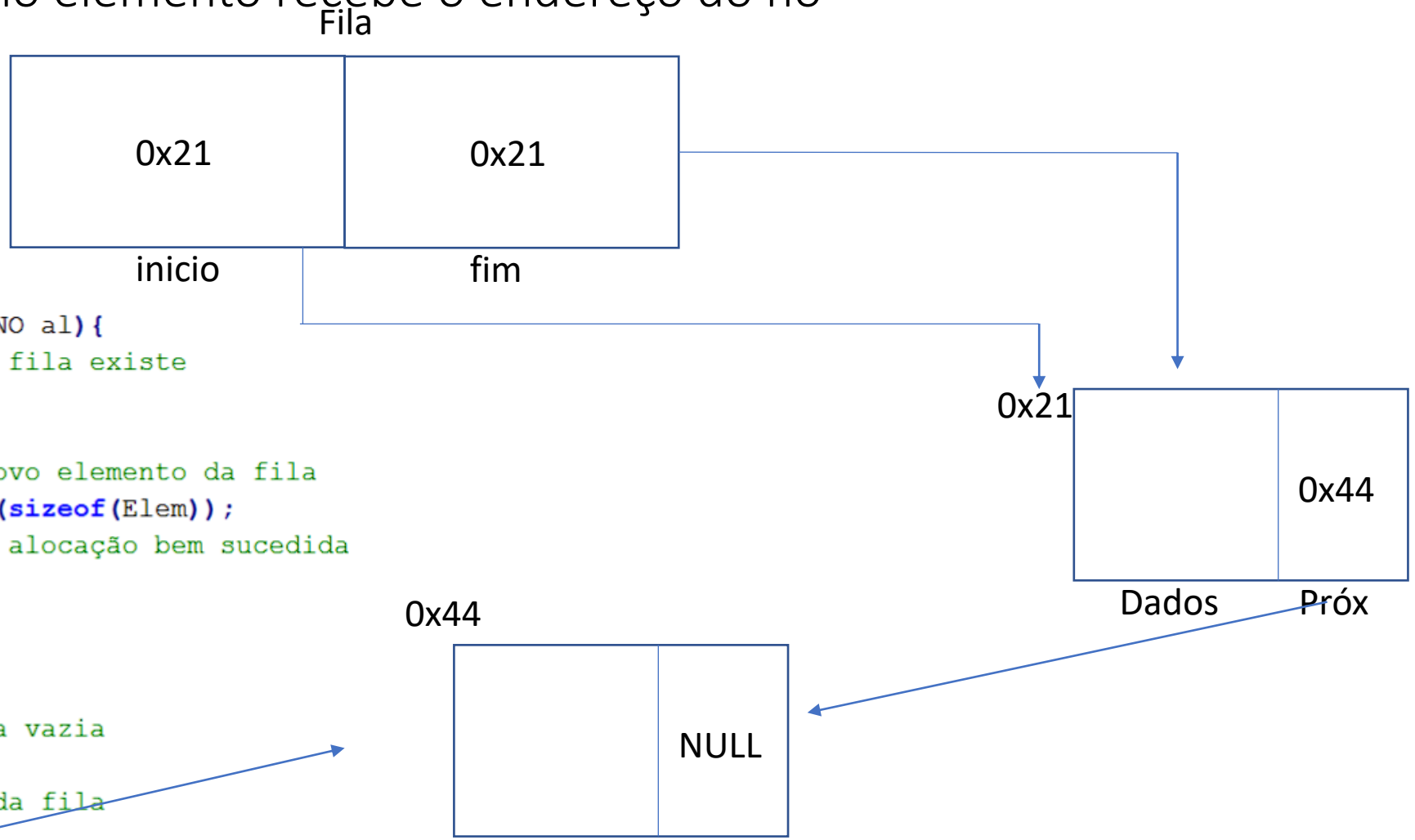
cria um nó recebendo os dados passados e apontando p/ null (é uma fila, logo o novo elemento será o último elemento da fila)




```
int insere_fila(Fila *fi, ALUNO al){
    if(fi == NULL){//Testa se fila existe
        return 0;
    }
    //aloca memória para um novo elemento da fila
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL){//testa se alocação bem sucedida
        return 0;
    }
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL){//fila vazia
        fi->inicio = no;
    }else{ //insere no final da fila
        fi->fim->prox = no;
    }
    fi->fim = no; //no passa a ser novo final da fila
    return 1;
}
```

Inserção na fila

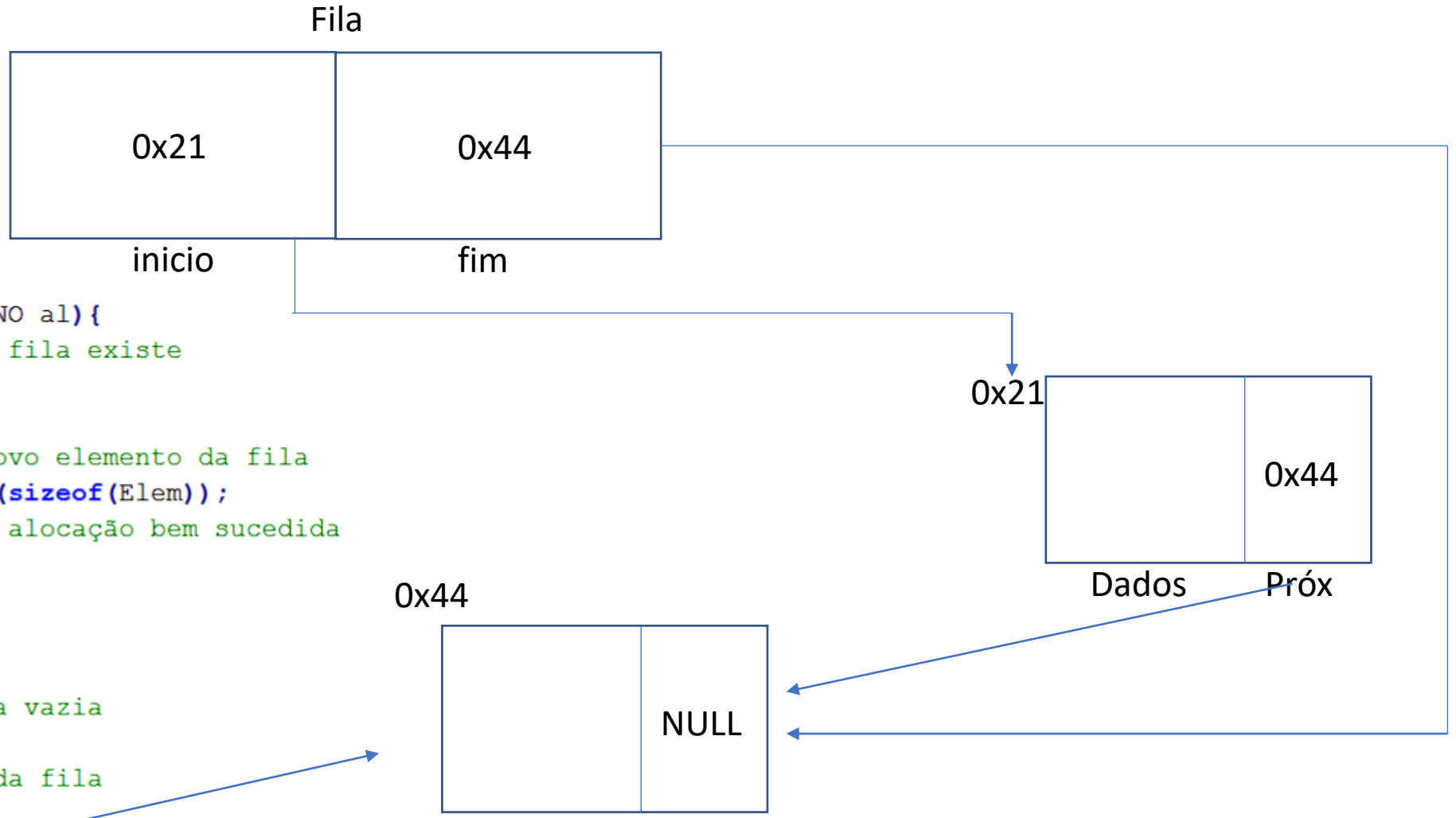
fim da fila é = NULL? Não, pois a fila não está vazia. Vamos cair no else. O próximo do último elemento recebe o endereço do nó




```
int insere_fila(Fila *fi, ALUNO al){
    if(fi == NULL){//Testa se fila existe
        return 0;
    }
    //aloca memória para um novo elemento da fila
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL){//testa se alocação bem sucedida
        return 0;
    }
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL){//fila vazia
        fi->inicio = no;
    }else{ //insere no final da fila
        fi->fim->prox = no;
    }
    fi->fim = no; //no passa a ser novo final da fila
    return 1;
}
```

Inserção na fila

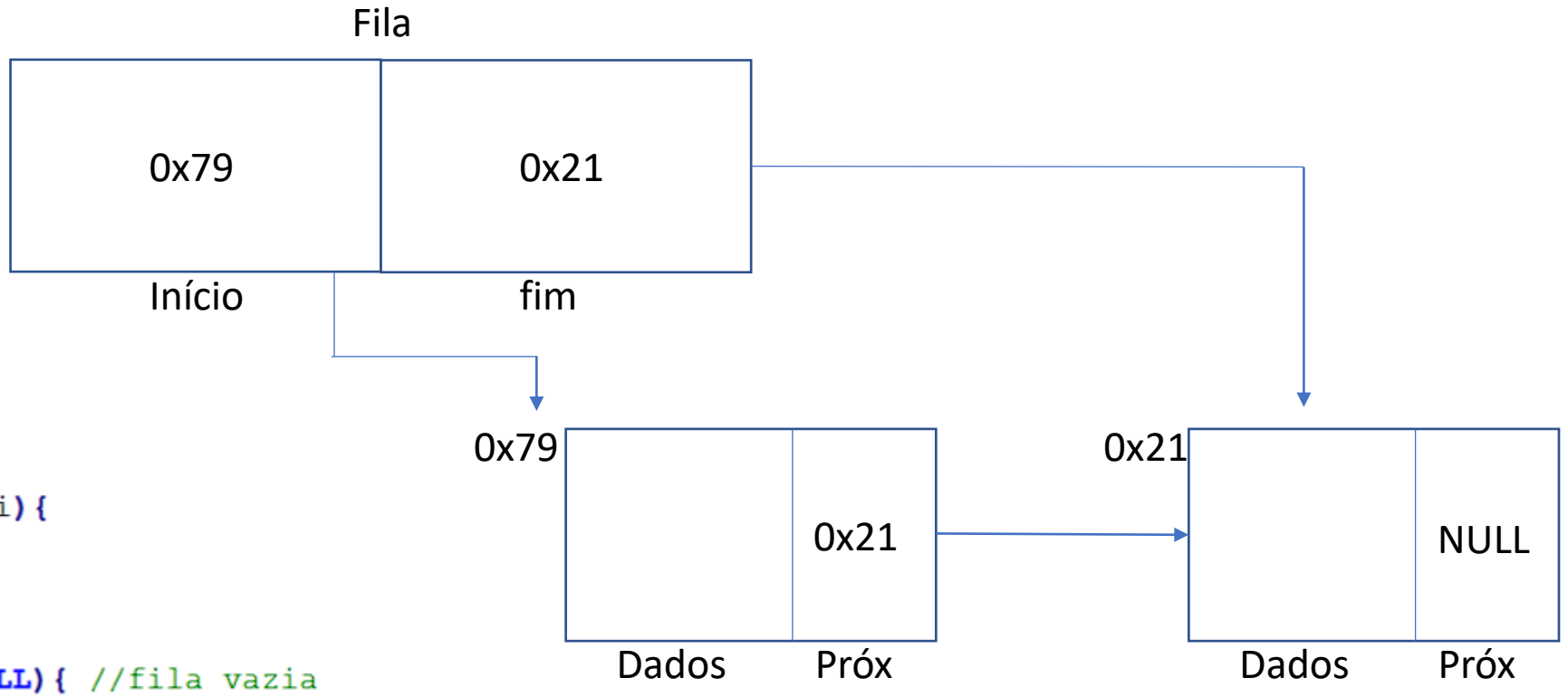
Agora temos que fazer o fim da fila apontar para o nó




```
int insere_fila(Fila *fi, ALUNO al){
    if(fi == NULL){//Testa se fila existe
        return 0;
    }
    //aloca memória para um novo elemento da fila
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL){//testa se alocação bem sucedida
        return 0;
    }
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL){//fila vazia
        fi->inicio = no;
    }else{ //insere no final da fila
        fi->fim->prox = no;
    }
    fi->fim = no; //no passa a ser novo final da fila
    return 1;
}
```

Remoção na fila

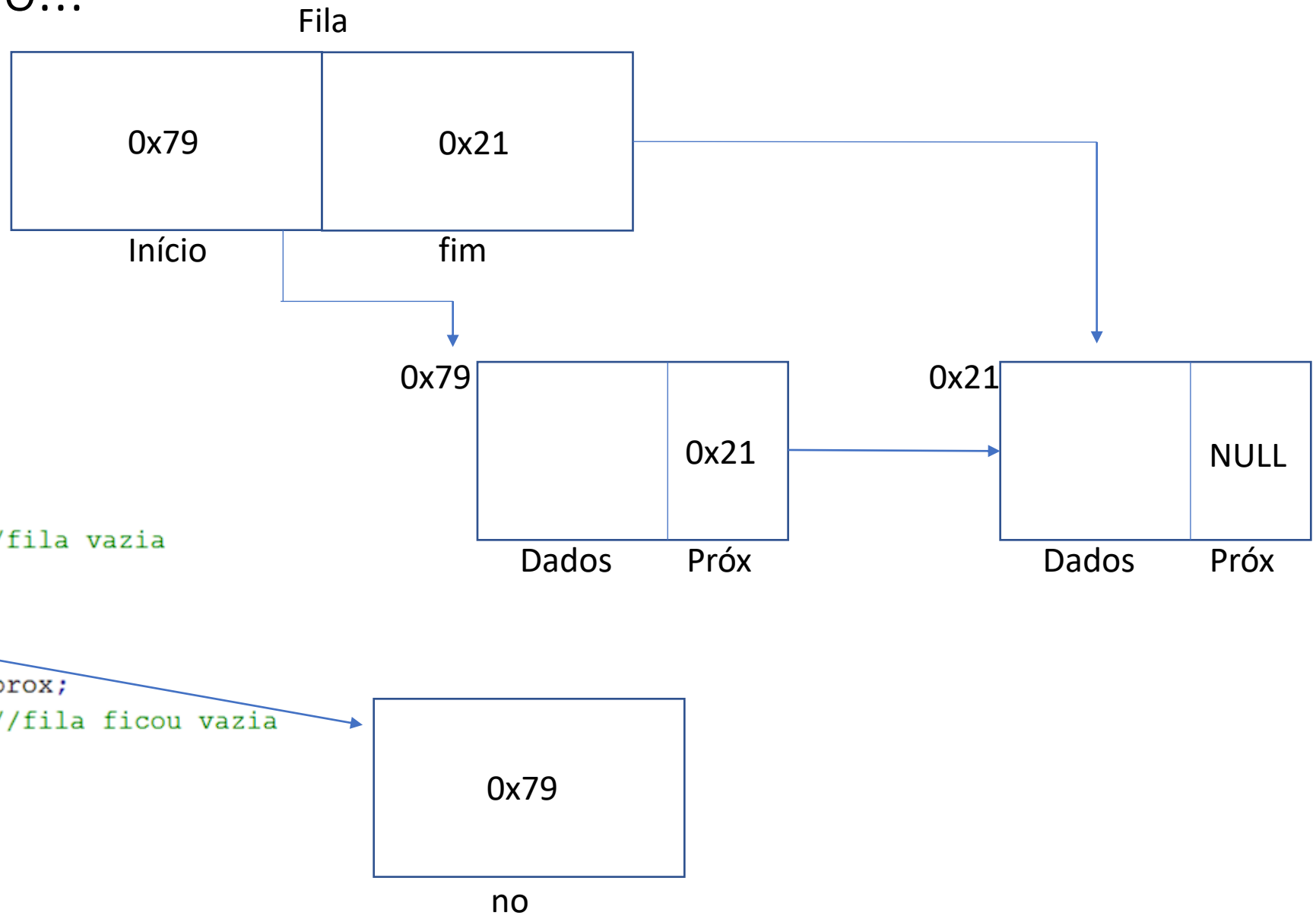
estado inicial




```c
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);
    return 1;
}
```

Remoção na fila

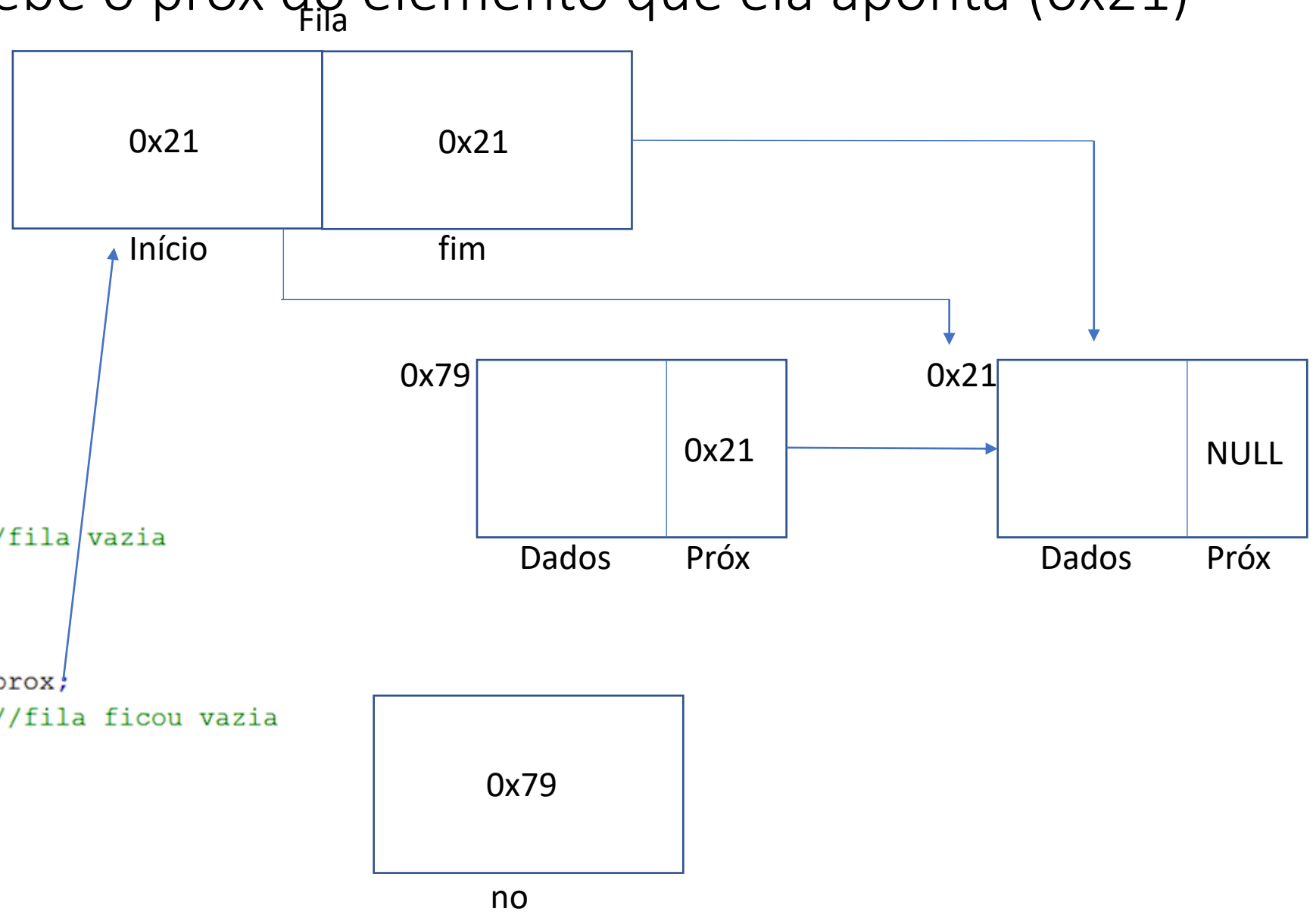
nó recebe inicio...




```c
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);
    return 1;
}
```

Remoção na fila

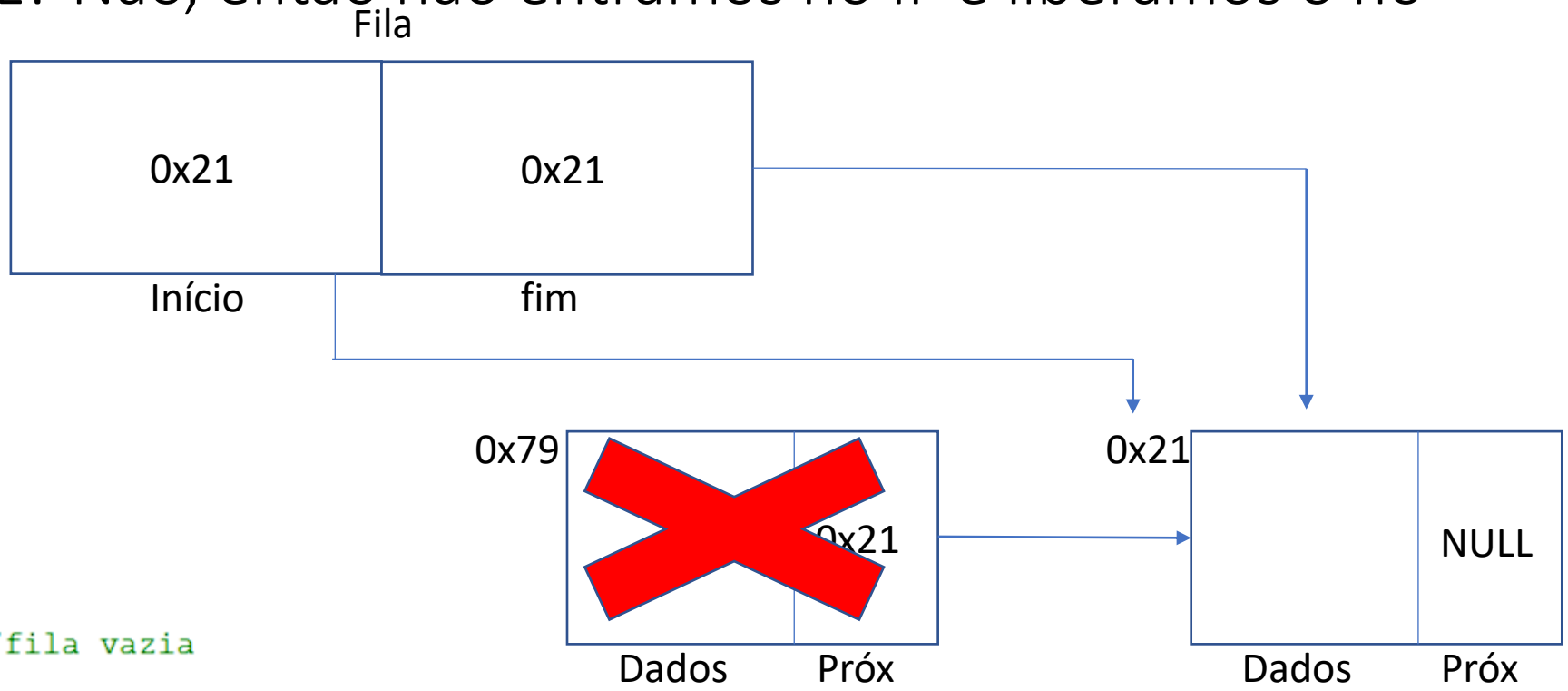
fila->inicio recebe o próx do elemento que ela aponta (0x21)




```c
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);
    return 1;
}
```

Remoção na fila

fila->inicio é NULL? Não, então não entramos no IF e liberamos o nó




```
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);
    return 1;
}
```