FaculdadeBarretos
www.faculdadebarretos.com.br

Algoritmos e Estruturas de Dados

(*Filas - Implementação e Uso*)

Prof. Me. Diogo Tavares da Silva
contato: *diogotavares@unibarretos.com.br*

Nas últimas aulas aprendemos...

- Listas encadeadas e seus métodos
- pilhas, métodos e manipulações

A estrutura de Fila (*Queue*)

- Também uma subclasse da estrutura de lista
  - Assim como a pilha (*stack*)
  - Na prática é uma lista encadeada...
  - PORÉM
    - Possui regras de inserção e remoção
      - **Primeiro a Entrar é o Primeiro a Sair**
        - **First In - First Out (FIFO)**
    - POR QUE?
      - Simular o comportamento da realidade!

A estrutura de Fila

- A estrutura de dados de fila é uma organização lógica de um comportamento natural no mundo real extremamente comum
  - Ex:
    - Fila de banco
    - Fila de mercado
    - linha de produção
    - ordem de atendimento no médico
    - fila pra pegar fila rsrs

A estrutura de Fila

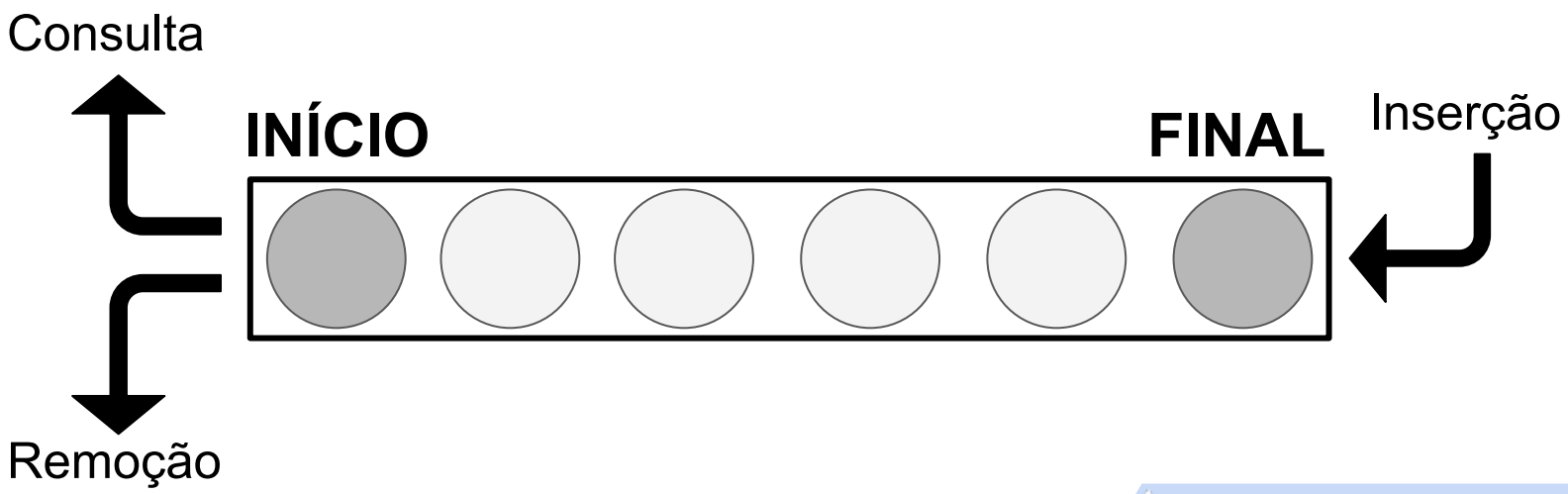
- Na computação são amplamente utilizadas
  - propósito:
    - Armazenar elementos e eventos que precisam ser armazenados, acessados e executados de acordo com a ordem de ocorrência no sistema.
    - Escalonamento de processos
    - Roteamento de pacotes
    - Atendimento de serviços
    - Algoritmos de árvores e grafos
      - Busca em largura, topsort, etc.
    - Simulação de eventos discretos
      - redes de **filas**

A estrutura de Fila

- Na computação são amplamente utilizadas
  - Exemplo:
    - **Simulação computacional**
      - Muitos problemas do mundo real seguem o comportamento de filas (**Teoria das filas**)
      - Simulação de eventos
        - Tarefas disparam eventos que vão entrando e saindo pelo sistema de filas que modela o sistema, obtendo-se estimativas

A estrutura de Fila

- Na prática:
  - Uma Fila convencional é uma lista encadeada em que a as inserções são realizadas por uma extremidade, o “final” , e as exclusões e consultas são feitas pela outra extremidade, o “início”.



Operações sobre filas

- Devem sempre respeitar a disciplina de acesso
  - Somente pelo início e final!
    - Se acessa, altera ou exclui nós abaixo do início **NÃO É FILA!**
    - Se insere em nós que não são o final, **NÃO É FILA!**
      - Cuidado na prova coleguinhas...

Operações sobre filas

- Deste modo, as seguintes operações são comuns para a estrutura de fila:
  - Criar a fila vazia
  - Verificar se a fila está vazia
  - Inserir um novo nó no final da fila (Enfileira ou *enqueue*)
  - Excluir o nó do início da fila (desenfileira ou *dequeue)*
  - Consultar e/ou modificar o nó que está no início da fila (consulta ou *first*)
  - Destruir a fila, liberando as posições reservadas para ela.

Projeto da estrutura do tipo fila

- Duas abordagens:
  1. Criar um tipo específico, assim como para pilha
  2. Usar um ponteiro que aponta para o início
     - Assim como o ponteiro na lista que aponta para a cabeça
     - PROBLEMA: Toda vez que vai inserir precisa que percorrer uma lista...

Projeto da estrutura do tipo fila

- Duas abordagens:
  1. Criar um tipo específico, assim como para pilha
  2. Usar um ponteiro que aponta para o início
     - Assim como o ponteiro na lista que aponta para a cabeça
     - PROBLEMA: Toda vez que vai inserir precisa que percorrer uma lista...

Projeto da estrutura nó de fila

- Deve conter a informação armazenada e um ponteiro para o próximo nó
  - Ou seja, um nó comum de lista.

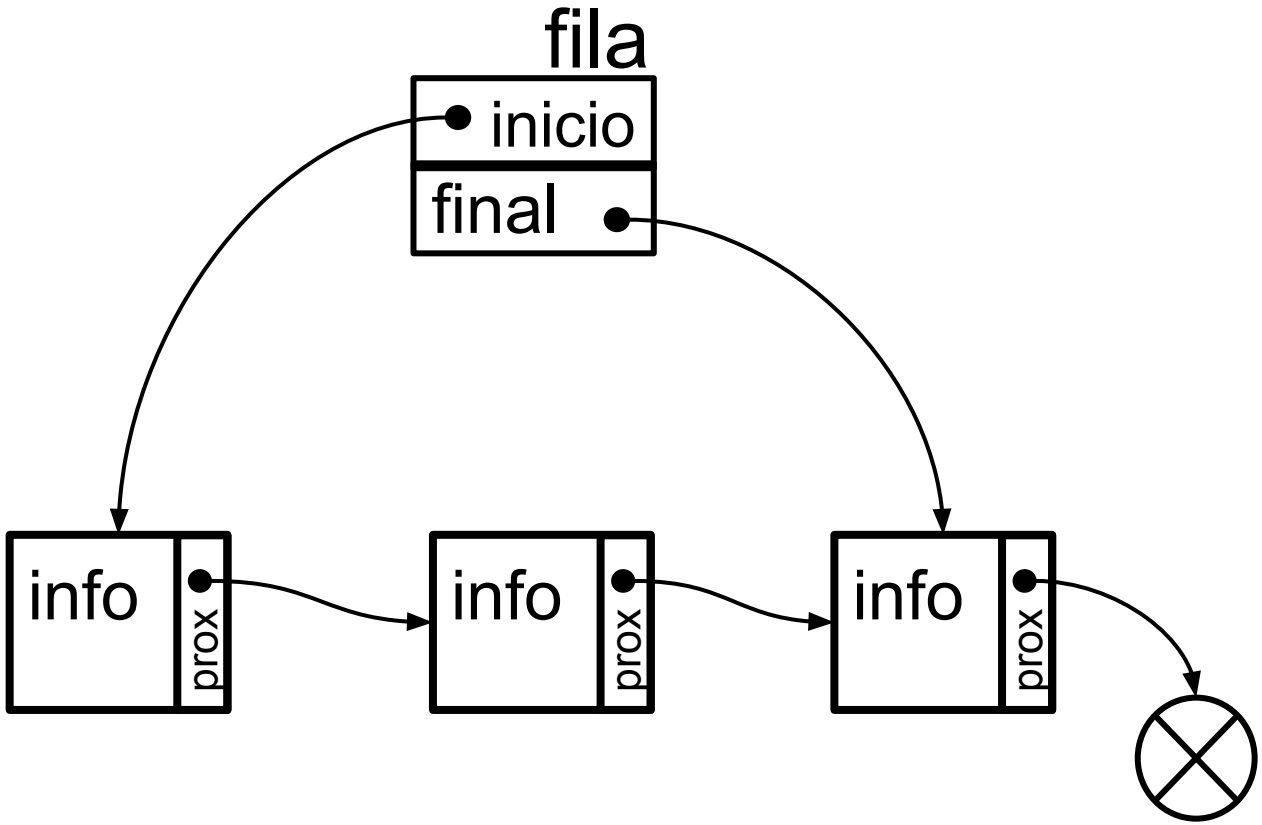
```
struct node{
   int info; //informação
   struct node *prox; //ponteiro para o próximo
conteúdo da fila
};
typedef struct node Node;
```

Projeto da estrutura de fila

- Após construir o nó:

```
struct fila{
   Nodo *inicio; //nó para o início da fila
   Nodo *final; //nó para o final da fila
}
typedef struct fila Fila;
```

Projeto da estrutura de fila

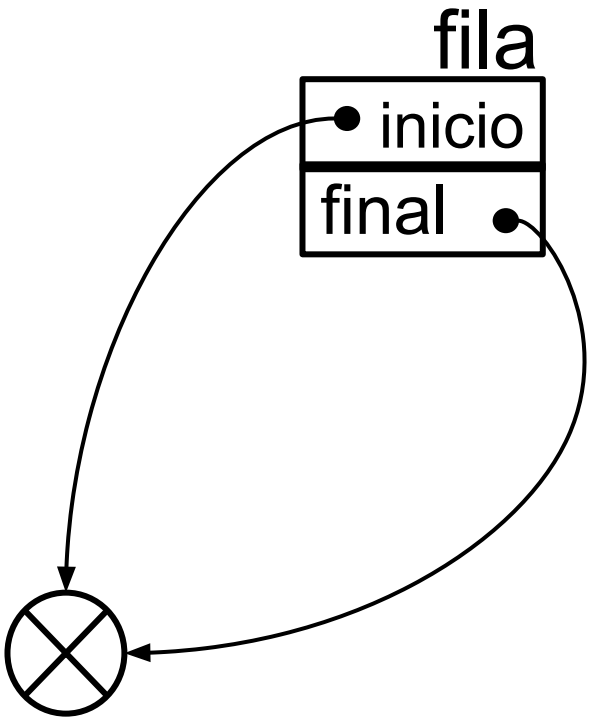


Criação da fila vazia

- Para criar a fila vazia é necessário criar uma função ***criaFila()***
  - Aloca o espaço da estrutura de fila na memória
  - Inicializa os ponteiros “início” e “final” apontando para NULL
    - A lista começa vazia.
  - Retorna um ponteiro para a estrutura de fila criada

Criação da fila vazia

- função ***criaFila()***



Criação da fila vazia

```
/*
FUNÇÃO: criaFila
RESUMO: Cria uma nova fila vazia
PARAM: void
RETORNO: Fila* (ponteiro para a fila criada)
*/
Fila* criaFila(){
    //armazena o espaço para a fila
    Fila* fila = new Fila;
    fila->inicio = NULL;
    fila->fim = NULL;
    return fila;
}
```

Função de teste *filaVazia()*

- Mais uma vez, criaremos uma função para verificar se a fila está vazia ou não
- Por essa razão, é conveniente criar uma função que testa se a lista está vazia
  - **int filaVazia(fila* fila)**
    - **retorna 1, se lista vazia**
    - **retorna 0, caso contrário**

Criação da função *filaVazia()*

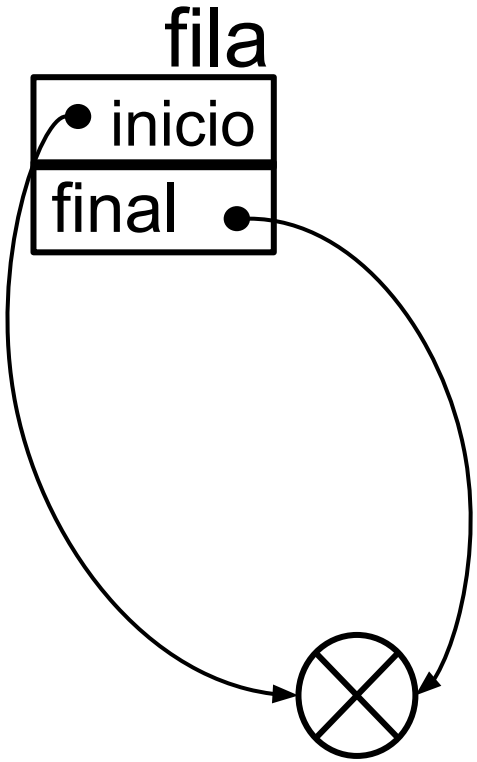
```
/*
FUNÇÃO: vazia
RESUMO: verifica se a fila está vazia
PARAM: Fila* fila (ponteiro para a fila)
RETORNO: int (valor de teste para a condição de vazio)
*/
int vaziaFila(Fila *fila){
    if(fila->inicio == NULL)
        return 1;
    else return 0;
}
```

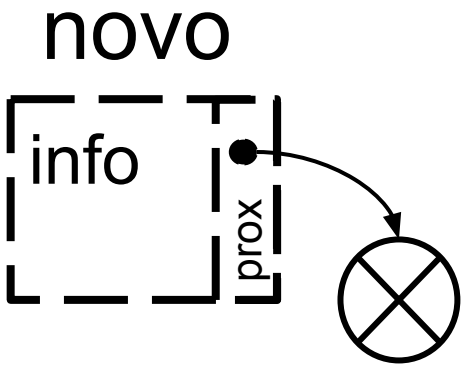
Função de enfileirar *(enqueue)*

- A função de enfileirar (*enqueue*) deve fazer uma inserção no final da fila.
  - Análogo a inserir no final da lista
    1. Cria-se no novo nó
    2. (fila->final)->prox = novo
    3. fila->final = novo

Função de emfileirar (enqueue)

- para inserir o primeiro nó

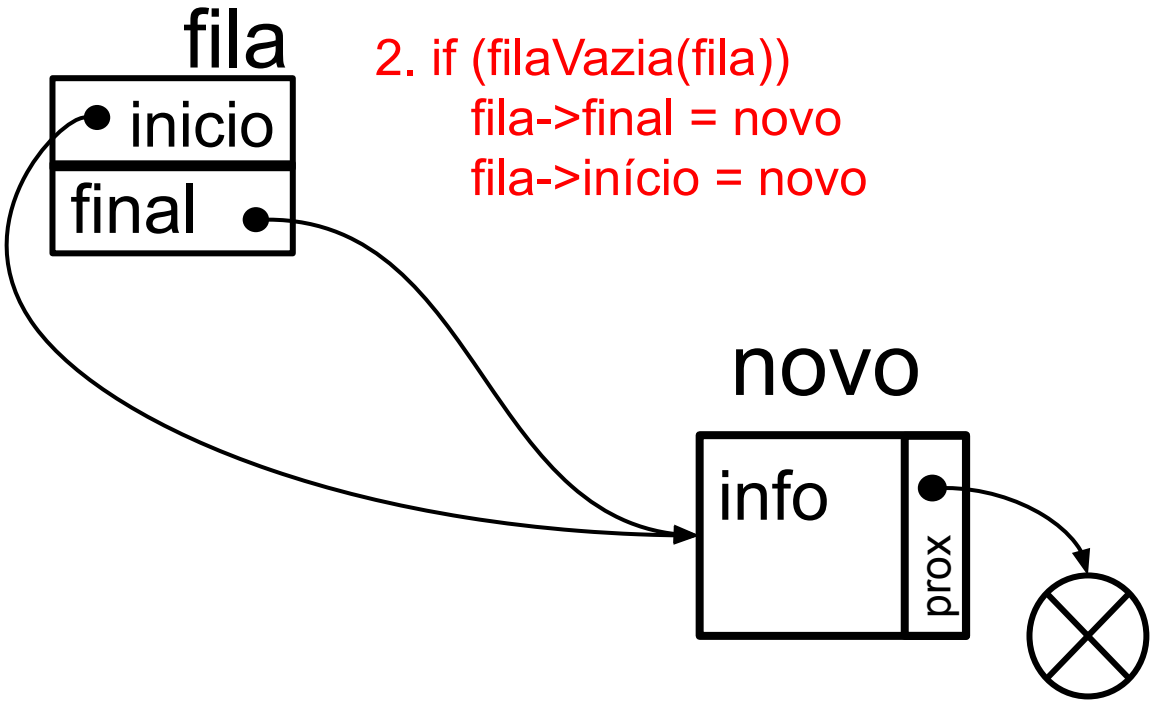




1. cria o novo nó

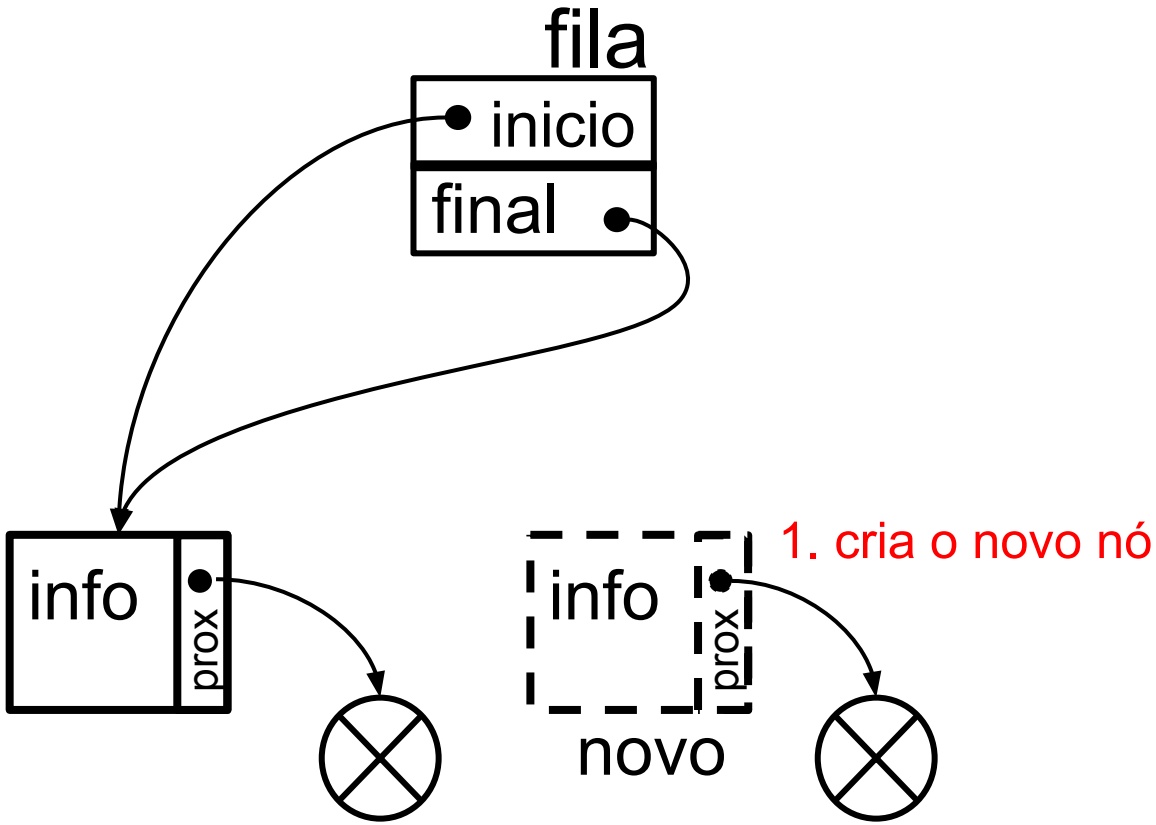
Função de emfileirar (enqueue)

- para inserir o primeiro nó



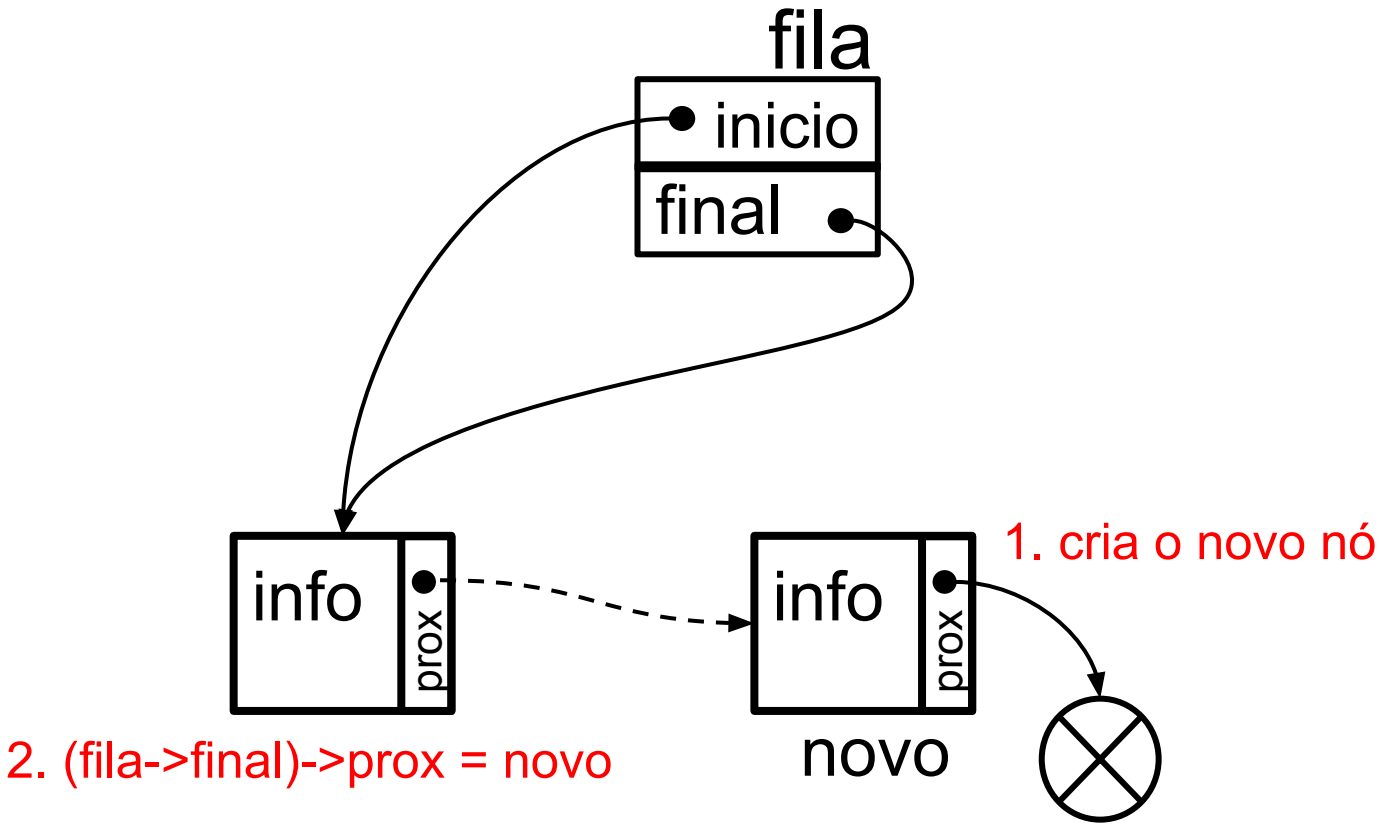
Função de emfileirar (enqueue)

- para inserir em fila não vazia



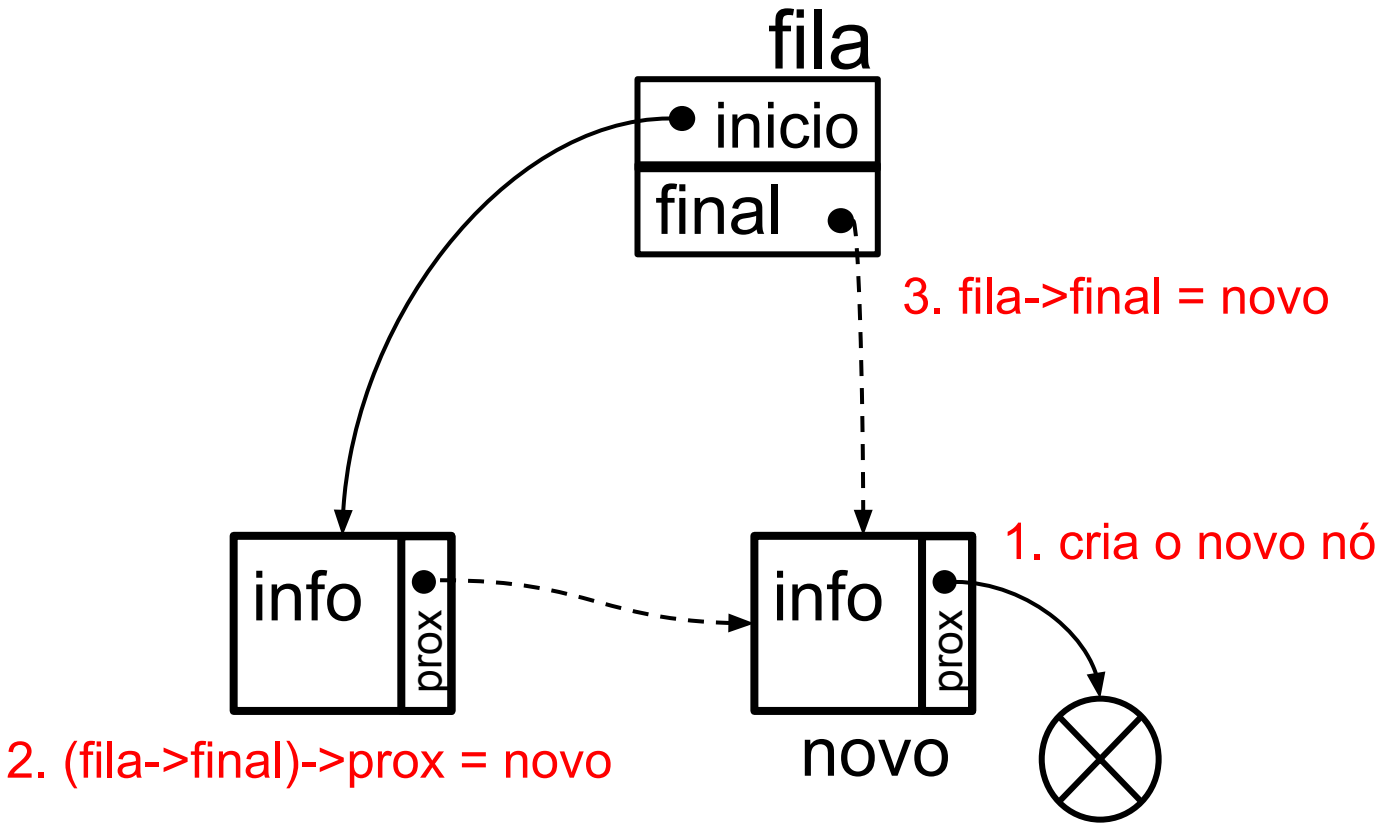
Função de emfileirar (enqueue)

- para inserir em fila não vazia



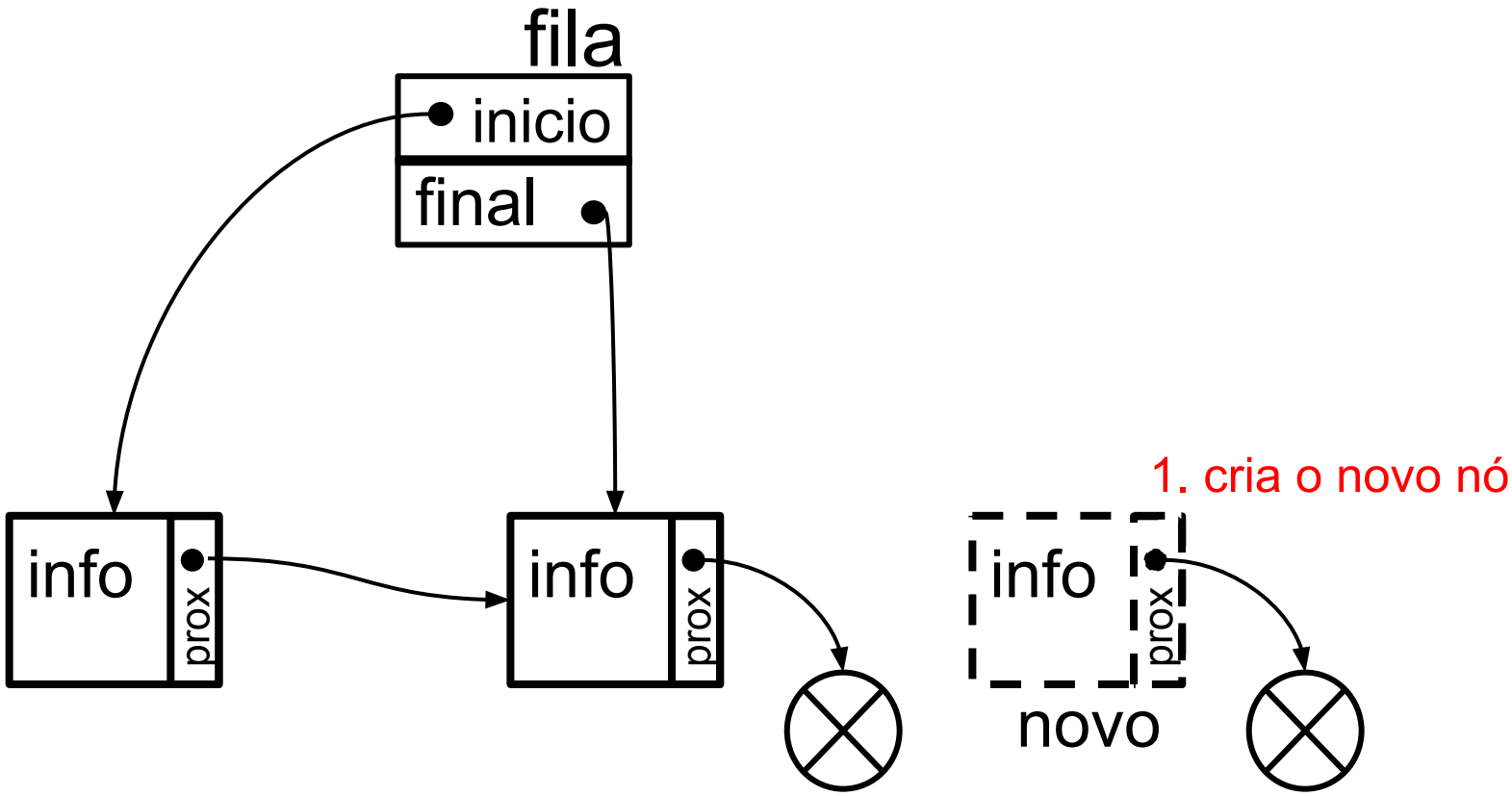
Função de emfileirar (enqueue)

- para inserir em fila não vazia



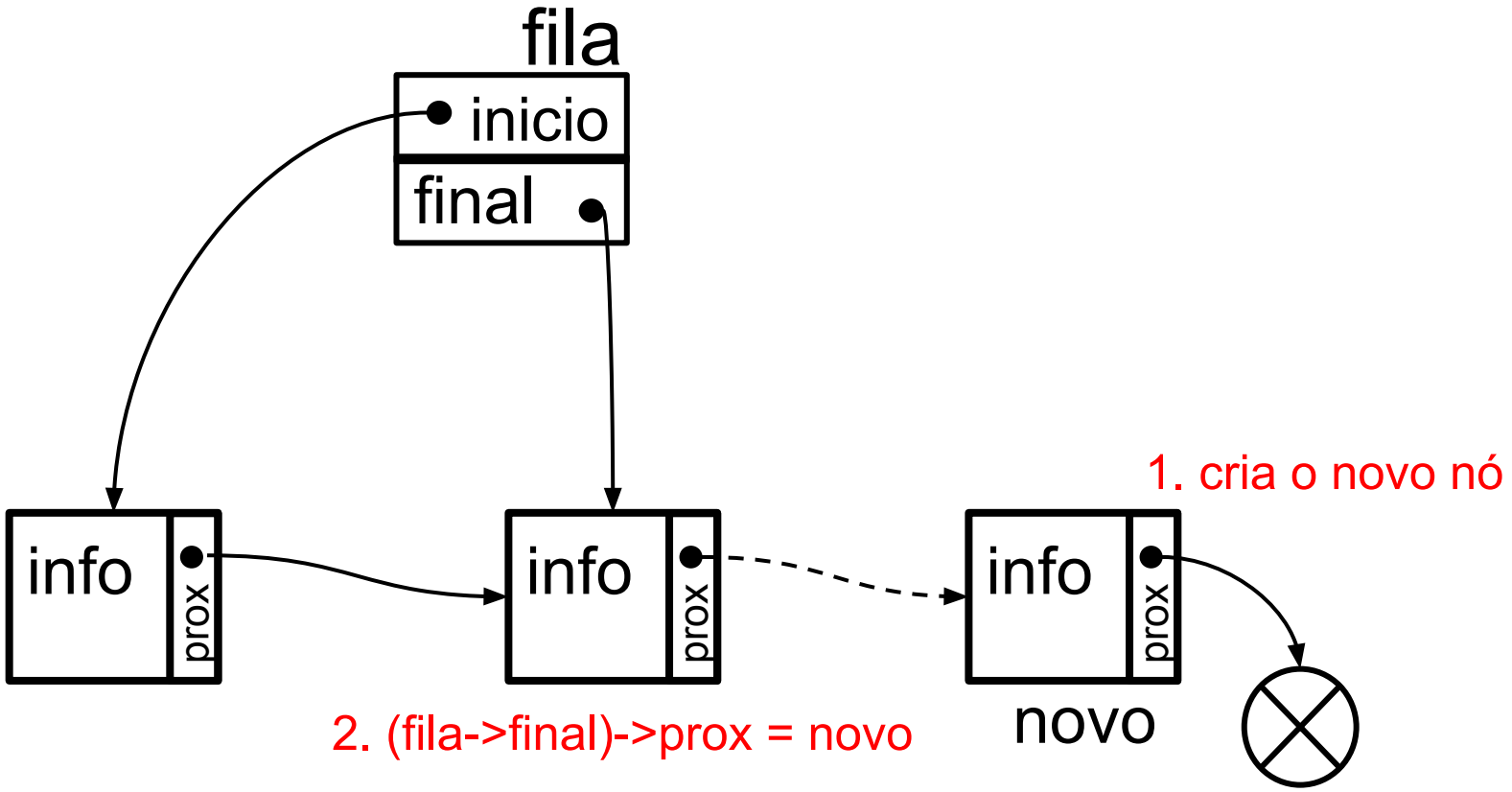
Função de emfileirar (enqueue)

- para inserir em fila não vazia



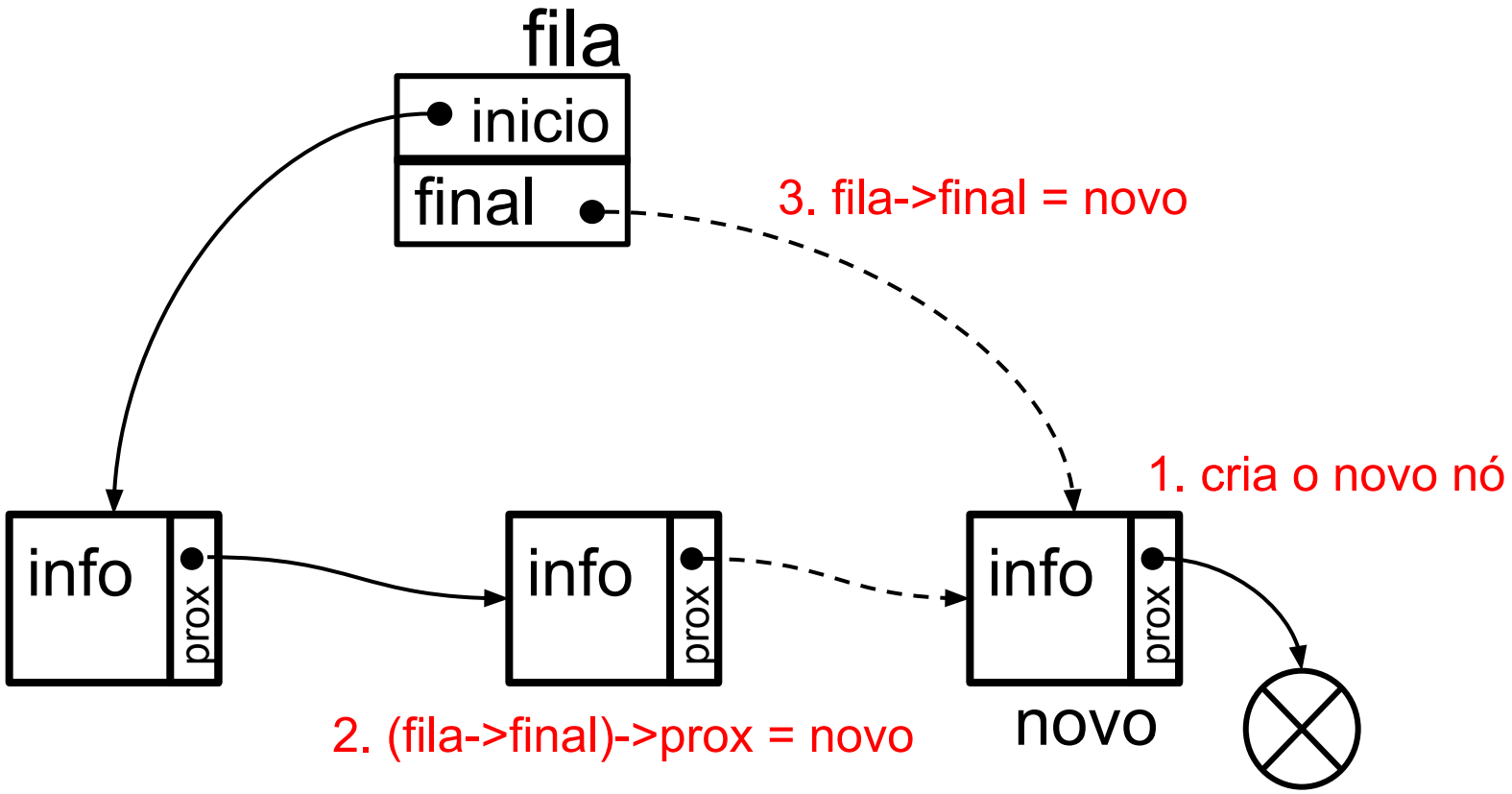
Função de emfileirar (enqueue)

- para inserir em fila não vazia



Função de emfileirar (enqueue)

- para inserir em fila não vazia



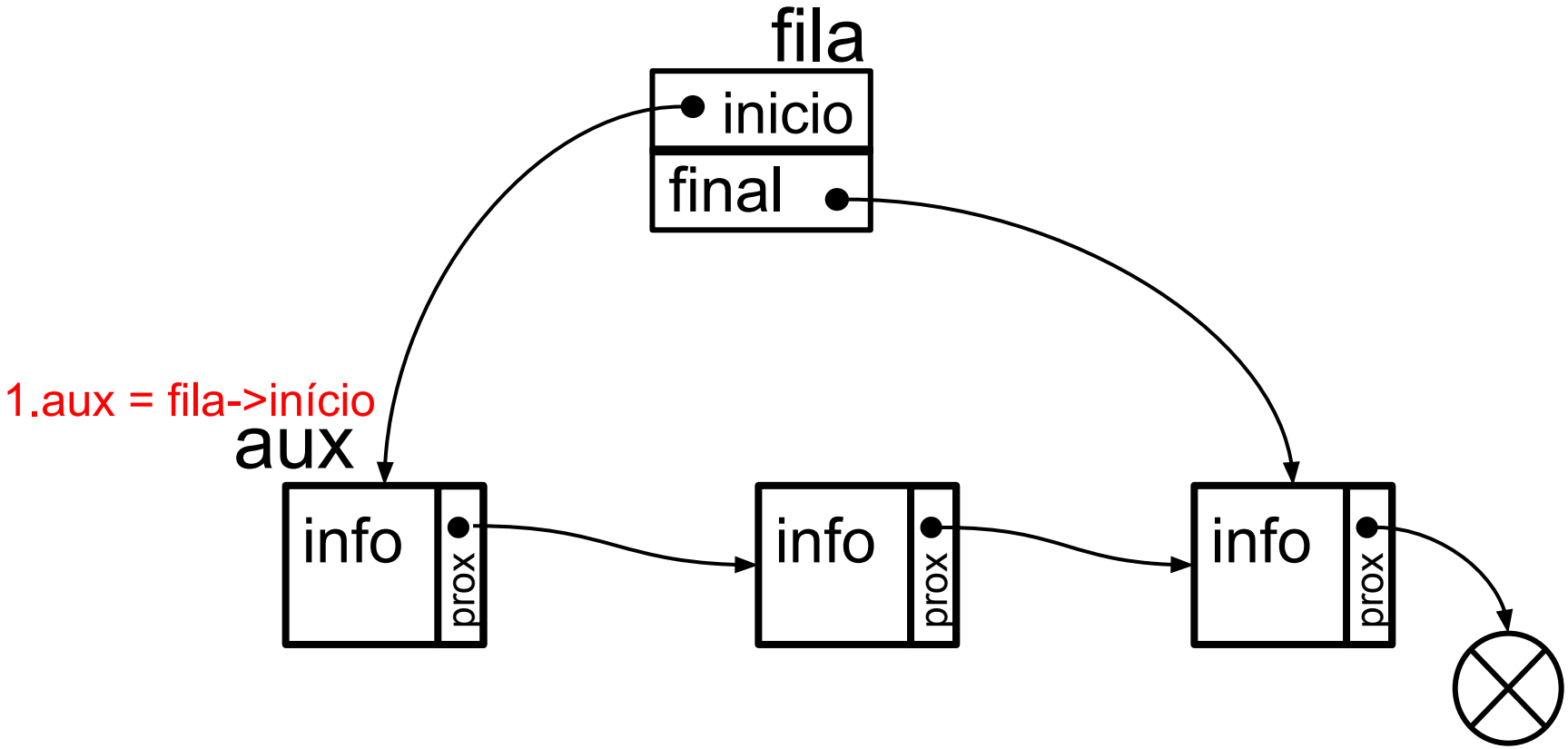
Função de enfileirar *(enqueue)*

```
/*
FUNÇÃO: enqueue
RESUMO: insere elemento no fim da fila
PARAM: Fila* fila (ponteiro para a fila), int valor (valor a ser inserido)
RETORNO: void
*/
void enqueue(Fila* fila, int valor){
    Node* novo = new Node;
    novo->info = valor;
    novo->prox = NULL; //inserido no final
    if(vaziaFila(fila)){
        fila->inicio = novo;
        fila->fim = novo;
    }else{
        (fila->fim)->prox = novo;
        fila->fim = novo;
    }
}
```

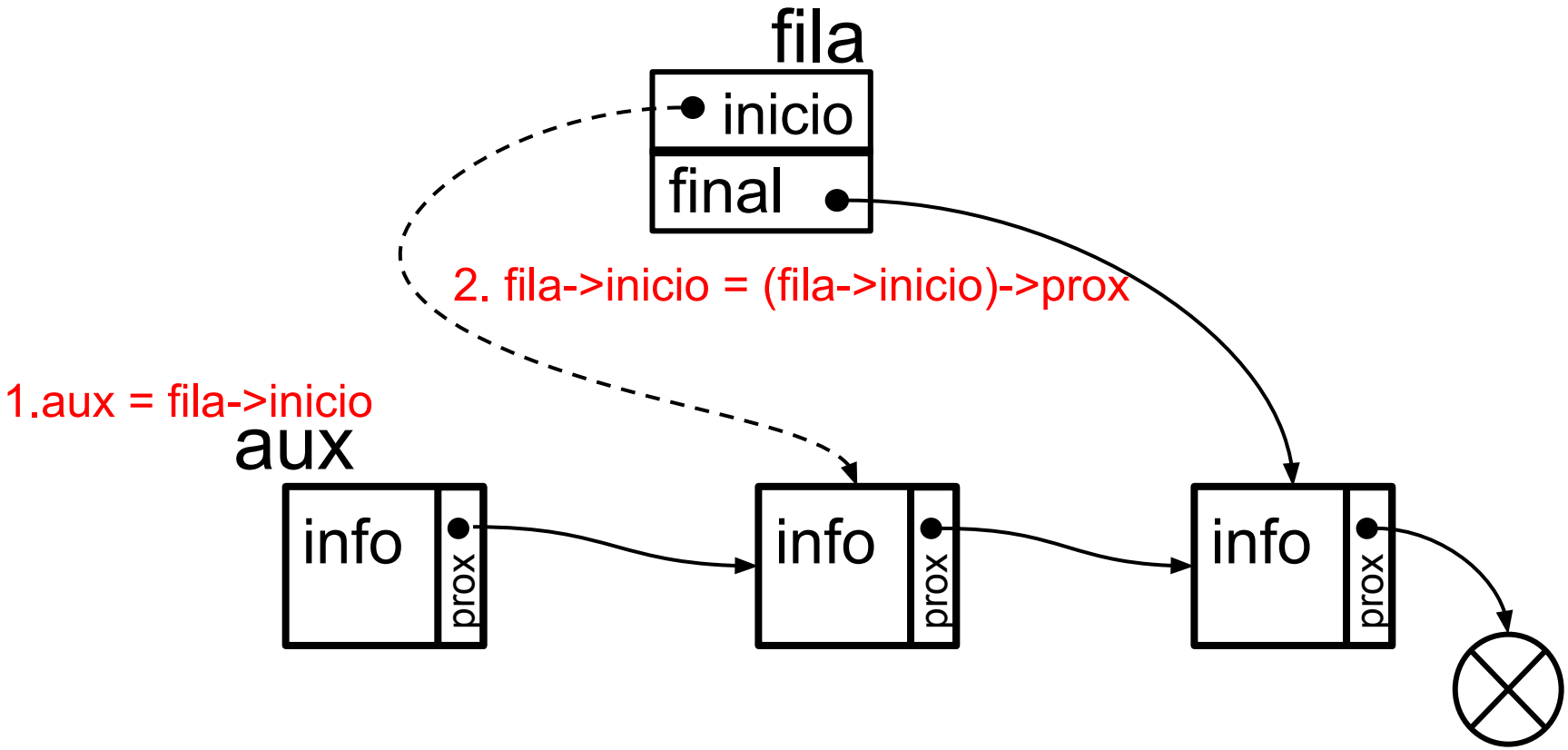
Função de desenfileirar *(dequeue)*

- A função de desenfileirar (*dequeue*) deve fazer uma remoção do elemento no início da fila retornando seu conteúdo.
  - Análogo a remover no início da lista
  - Se a fila não é vazia
    1. aux ← fila->início
       1.1. valor = aux->info;
    2. fila->início ← (fila->início)->prox
    3. liberar (aux)
    4. retornar valor;

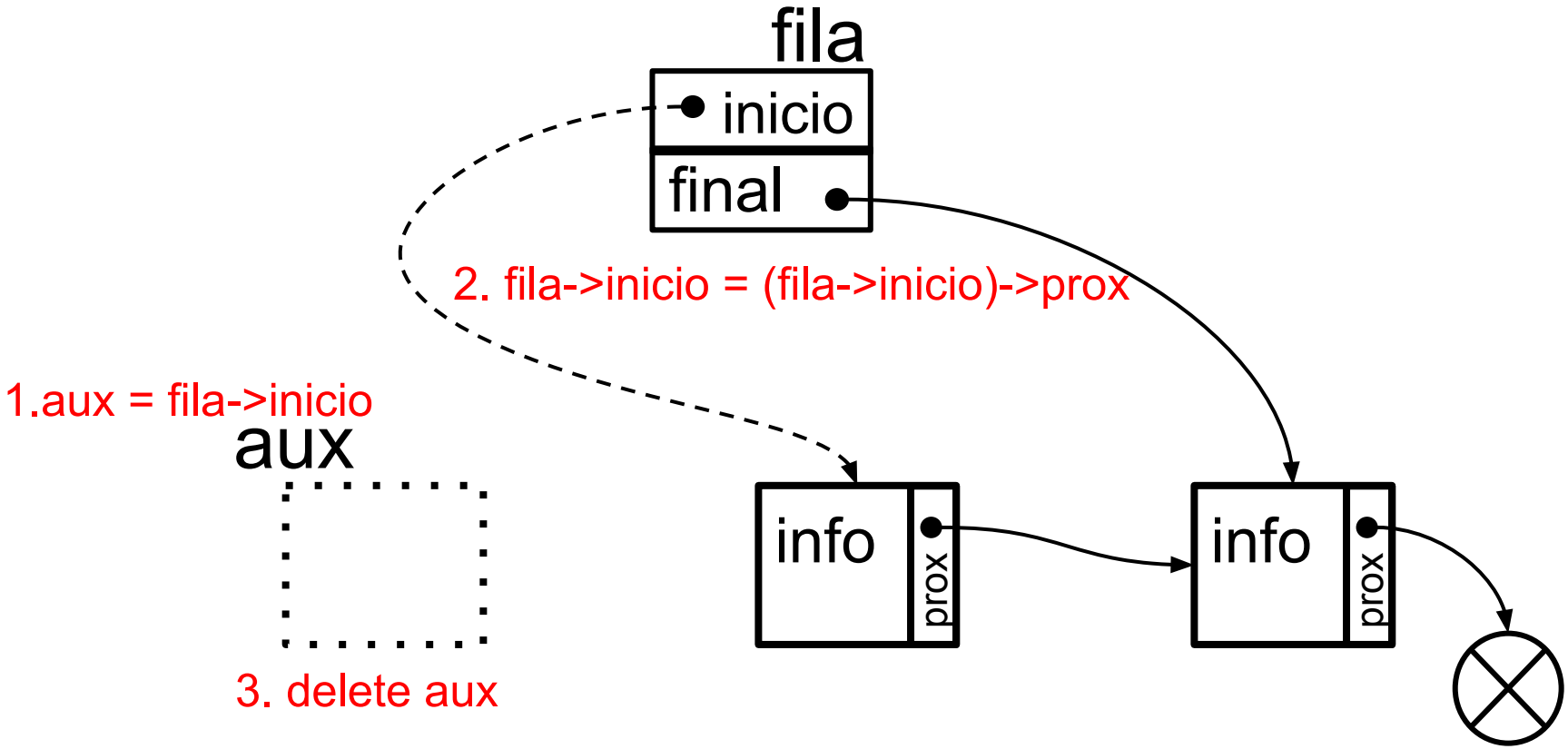
Função de desenfileirar (*dequeue*)



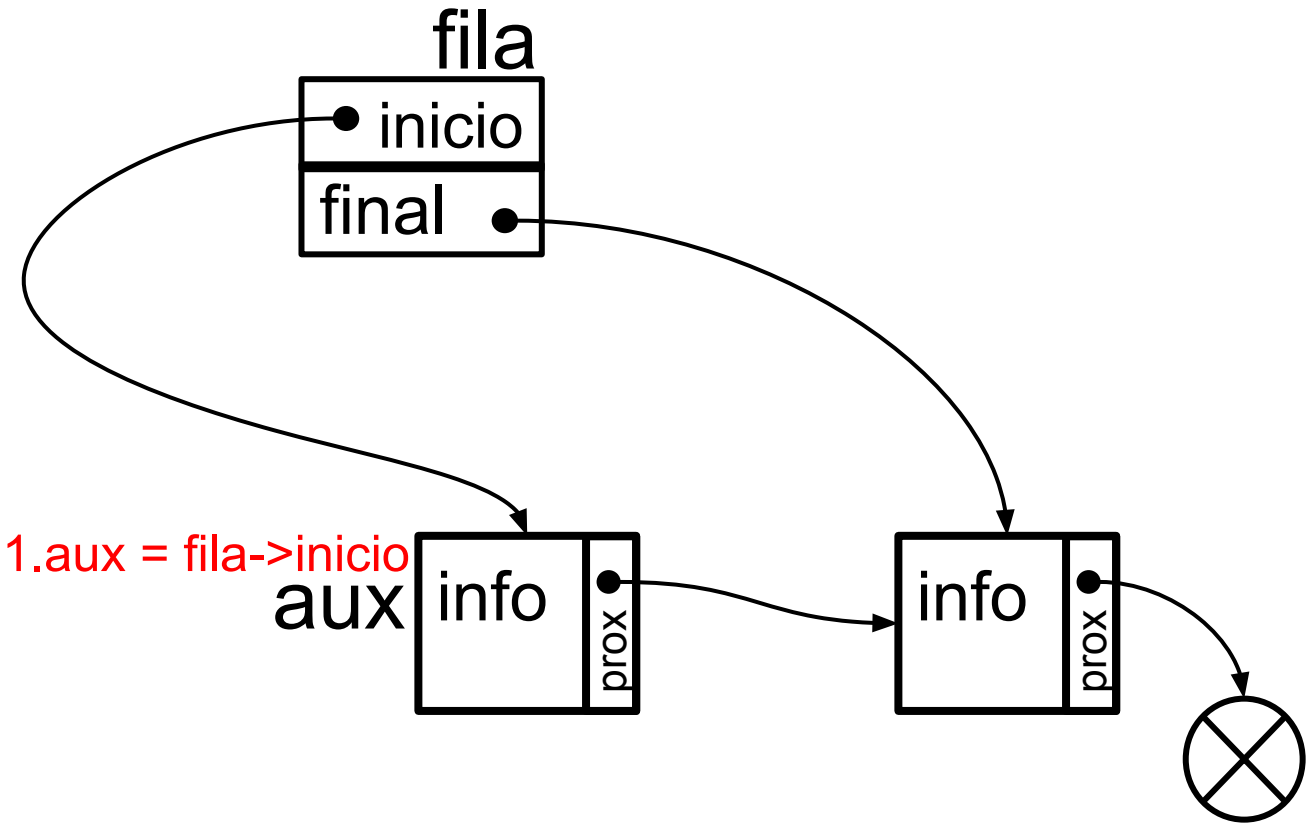
Função de desenfileirar (*dequeue*)



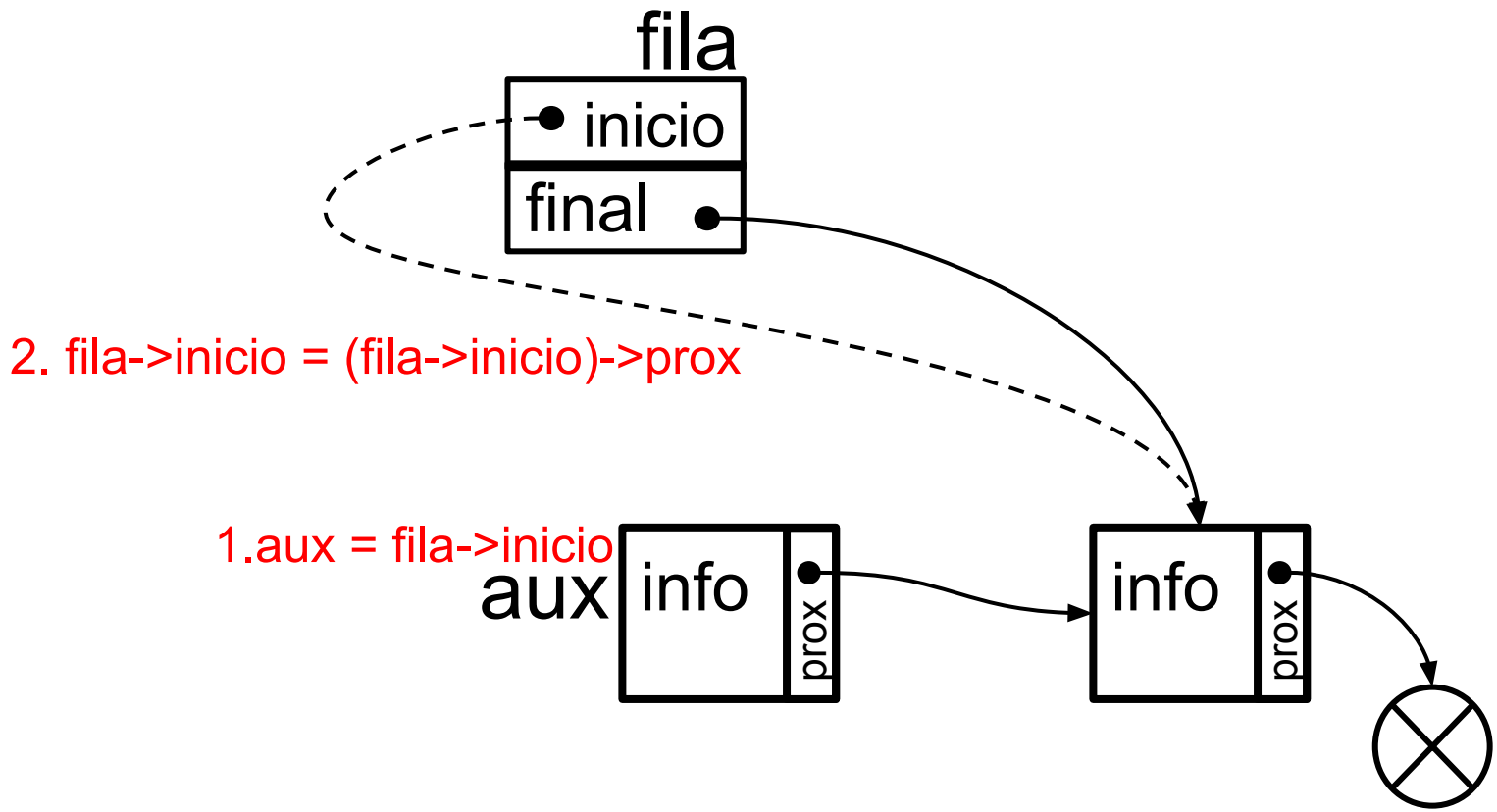
Função de desenfileirar (*dequeue*)



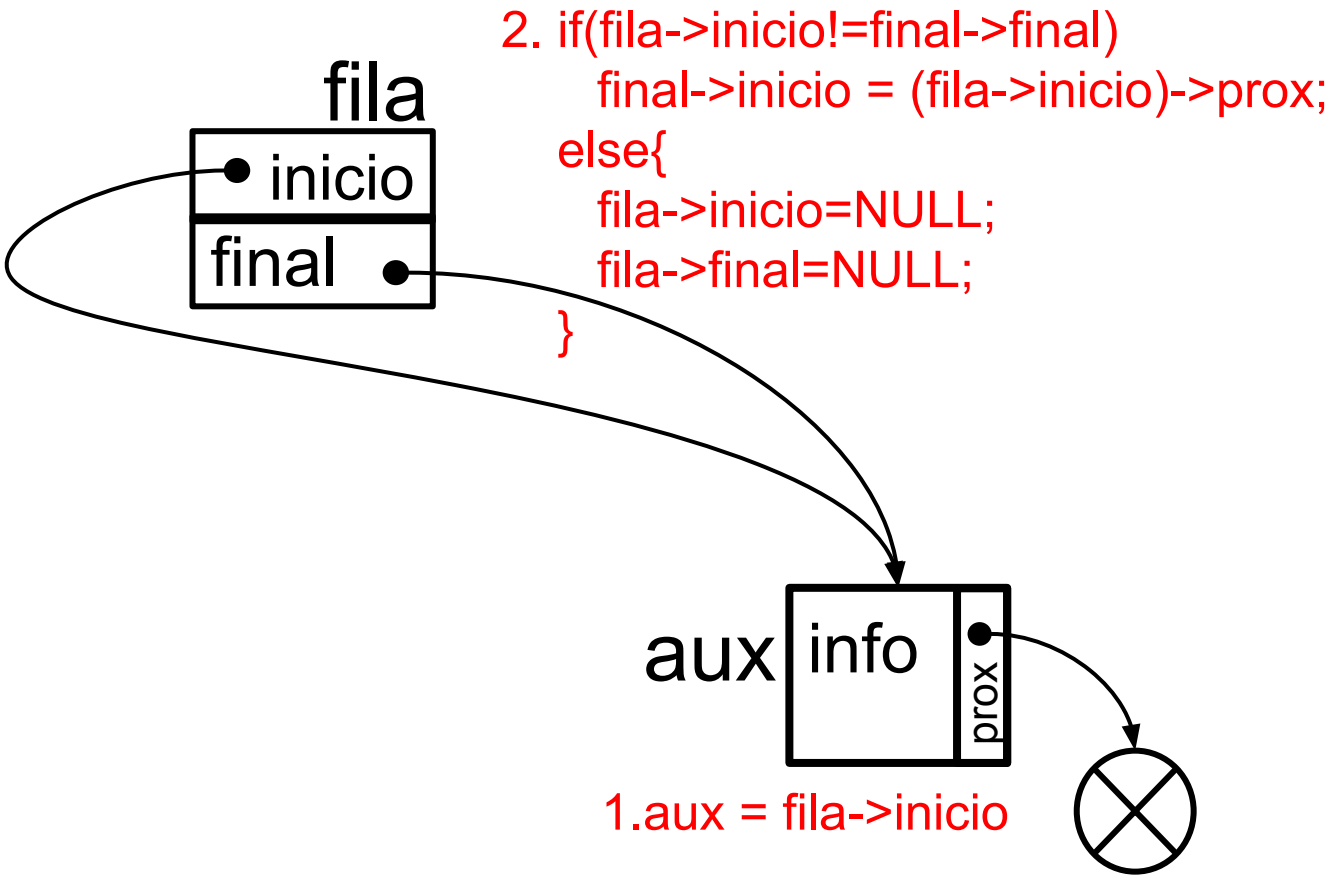
Função de desenfileirar (*dequeue*)



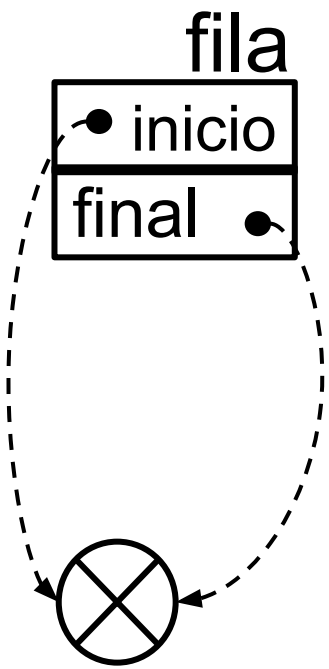
Função de desenfileirar (*dequeue*)



Função de desenfileirar (*dequeue*)

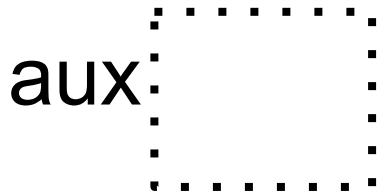


Função de desenfileirar (*dequeue*)




```
2. if(fila->inicio!=final->final)
     final->inicio = (fila->inicio)->prox;
   else{
     fila->inicio=NULL;
     fila->final=NULL;
   }
```

3. delete aux



1.aux = fila->inicio

Função de desemfileirar *(dequeue)*

```
/*
FUNÇÃO: dequeue
RESUMO: remove e retorna o elemento do início da fila
PARAM: Fila* fila (ponteiro para a fila)
RETORNO: int (valor removido da fila)
*/
int dequeue(Fila* fila){
    Node* aux;
    int valor;
    aux = fila->inicio;
    valor = aux->info;
    fila->inicio = aux->prox;
    //se esvaziou a fila, fila->fim tbm tem que ficar nula
    if(vaziaFila(fila))
        fila->fim = NULL;
    delete(aux);
    return valor;
}
```

Função de consultar início *(first)*

- A função de consultar o conteúdo do início da fila (*first*) deve retornar o conteúdo do início da fila sem alterá-lo.
  - Se a fila não é vazia
    1. retornar (fila->inicio)->info;

Função consultar início *(first)*

```
/*
FUNÇÃO: first
RESUMO: retorna o elemento do início da fila
PARAM: Fila* fila (ponteiro para a fila)
RETORNO: int (valor da cabeça da fila)
*/
int first(Fila* fila){
    if (!vaziaFila(fila))
        return (fila->inicio)->info;
}
```

Função de destruir fila *(destroiFila)*

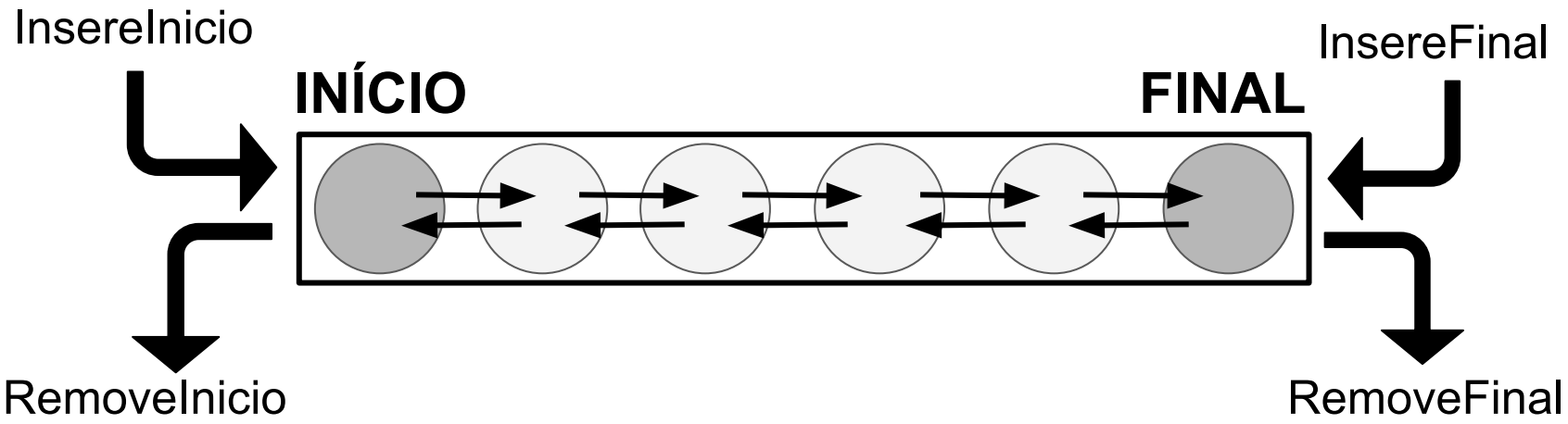
- Respeitando as regras de fila SEMPRE!
  - removendo pelo início até esvaziar
- Enquanto a fila não é vazia
  - dequeue(fila);

Operações com Filas (Trabalho)

- Assim como com as pilhas, consiste sempre em enfileirar e desenfileirar
- Exercícios para vocês
    1. Imprimir uma fila
    2. Buscar e editar um elemento da fila
    3. Buscar e remover um elemento da fila
    4. Remover todas as repetições da fila
    5. remover todos os pares da fila

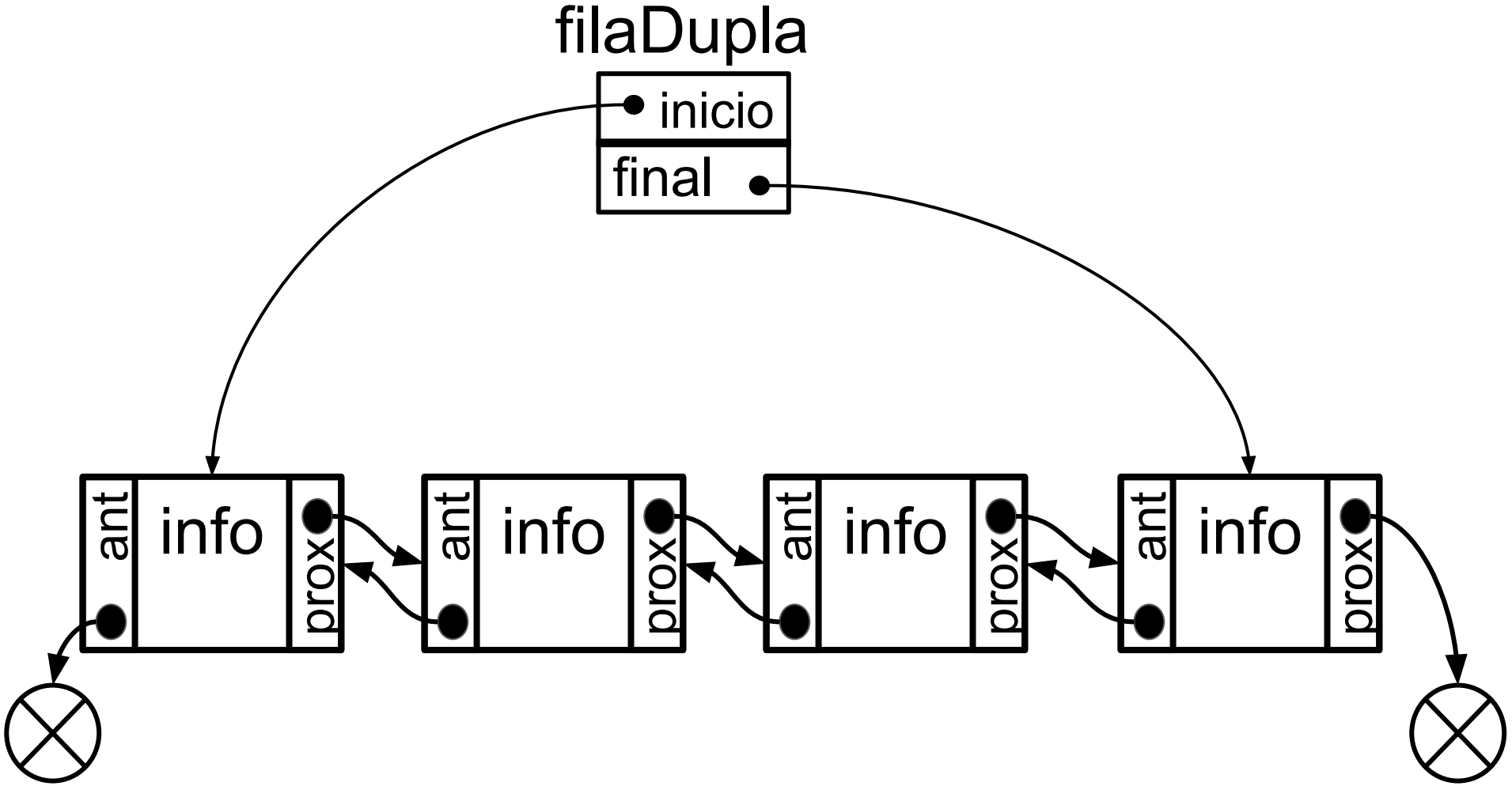
Variantes de Fila

- **Fila dupla (*deque*)**
  - Implementada com uma lista duplamente encadeada
    - Permite inserções e remoções tanto pelo início quanto pelo final
    - Se comporta tanto como **fila**, tanto como **pilha**



Variantes de Fila

- **Fila dupla**



Variantes de Fila

- **Fila Circular**
  - Implementada com uma lista circular simplesmente ou duplamente encadeada
    - Aplicações:
      - Eventos que são atendidos e voltam a ser inseridos nas filas (eventos cíclicos)
        - ex:
          - (escalonamento circular - Round Robin)

Variantes de Fila

- **Fila com prioridade**
  - Além das regras convencionais da fila (FIFO)
  - Adição de uma prioridade de atendimento
    - campo na estrutura que indica a prioridade, por exemplo
    - exemplos:
      - Atendimento prioritário dos bancos, mercados, ônibus, etc.
      - atendimento prioritário no hospital do idoso
        - prioridade por idade
      - Vips da balada, rsrs
      - Processos do admin executados antes de processos de usuário
      - Níveis de prioridade entre processos, requisições, ordens de serviço, etc.

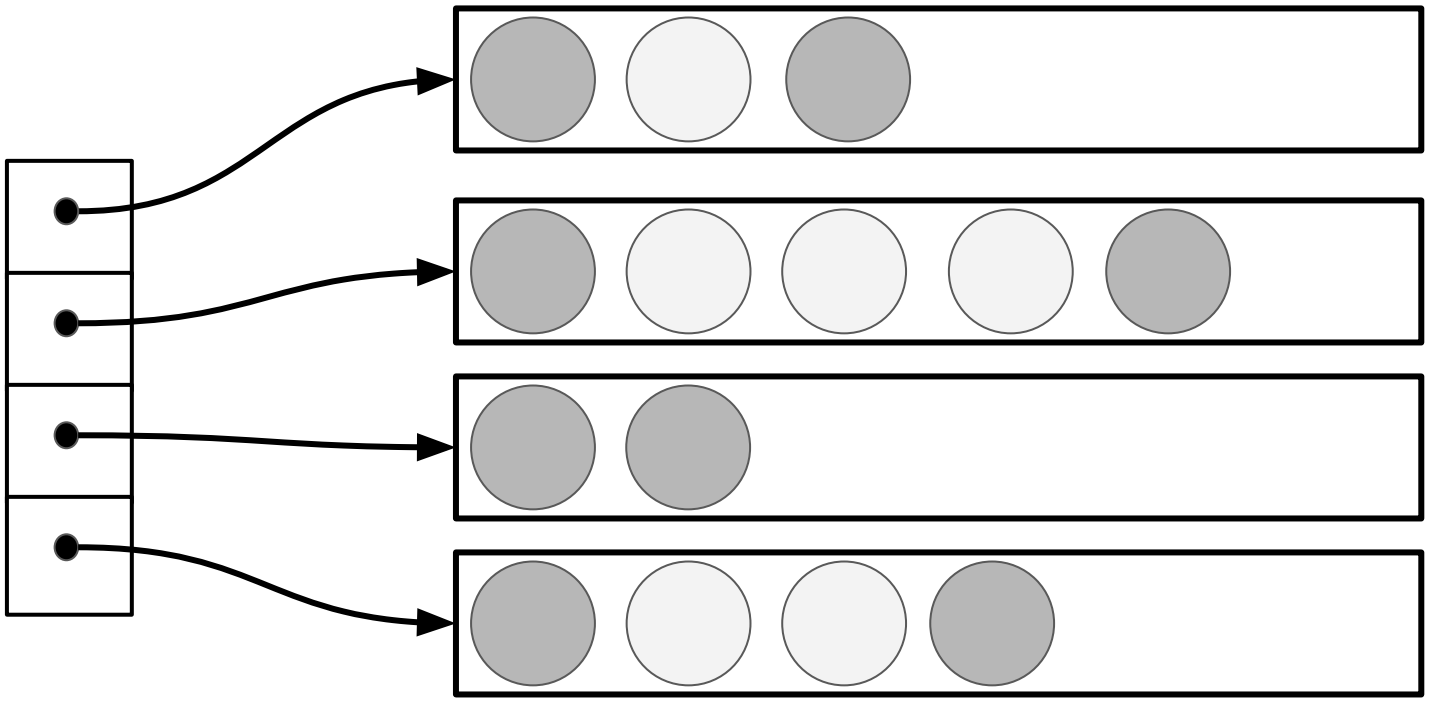
Fila com prioridade - projeto de estrutura

- Um possível projeto de nó seria incluir um campo “pri” que indica o nível de prioridade.
  - Inserção pela ordem de prioridade e em seguida de chegada
  - Remoção normalmente pelo início

```
struct nodePri{
   int info; //informação
   int pri; //prioridade
   struct node *prox; //ponteiro para o próximo conteúdo da fila
};
typedef struct node NodePri;
```

Fila com prioridade - projeto de estrutura

- Outra abordagem possível, é estabelecer níveis fixos de prioridade.
  - Cada nível é o índice para um vetor de filas



Fila *vet[4]; *//4 níveis de prioridade*
Fila *vet = malloc(sizeof(Fila*)*4);