# ALGORITMOS E ESTRUTURAS DE DADOS I

Prof. Caio César de Freitas Dantas

# Filas

- Tipo Abstrato de Dados com a seguinte característica:
    - O primeiro elemento a ser inserido é o primeiro a ser retirado.
    - FIFO - First in First Out.
- Analogia: fila bancária, fila do cinema, etc.
- Usos: Sistemas operacionais: fila de impressão, fila de processamento, etc.

# Filas

- Fila nada mais é do que uma Lista com uma restrição:
    - O primeiro elemento a ser inserido é o primeiro a ser retirado.

- O que o TAD Fila deveria conter?
    - Representação do tipo da fila.
    - Conjunto de operações que atuam sobre a fila.

- Quais operações deveriam fazer parte da fila?
    - Depende de cada aplicação.
    - Mas, um conjunto padrão pode ser definido.

# Filas

Operações necessárias à grande maioria das aplicações:

- Fila_Inicia(Fila): inicia uma fila vazia.
- Fila_Enfileira(Fila, x): insere o item x no final da fila.
- Fila_Desenfileira(Fila, x): retorna o item x no início da fila, retirando-o da fila.
- Fila_EhVazia(Fila): retorna true se a fila está vazia, e false caso contrário.

# Filas

- Existem várias opções de estruturas de dados que podem ser usadas para representar filas.

- As duas representações mais utilizadas são:

    - Implementação por arrays.
    - Implementação por ponteiros.

# Filas

TAD Fila: Implementação por ARRAY

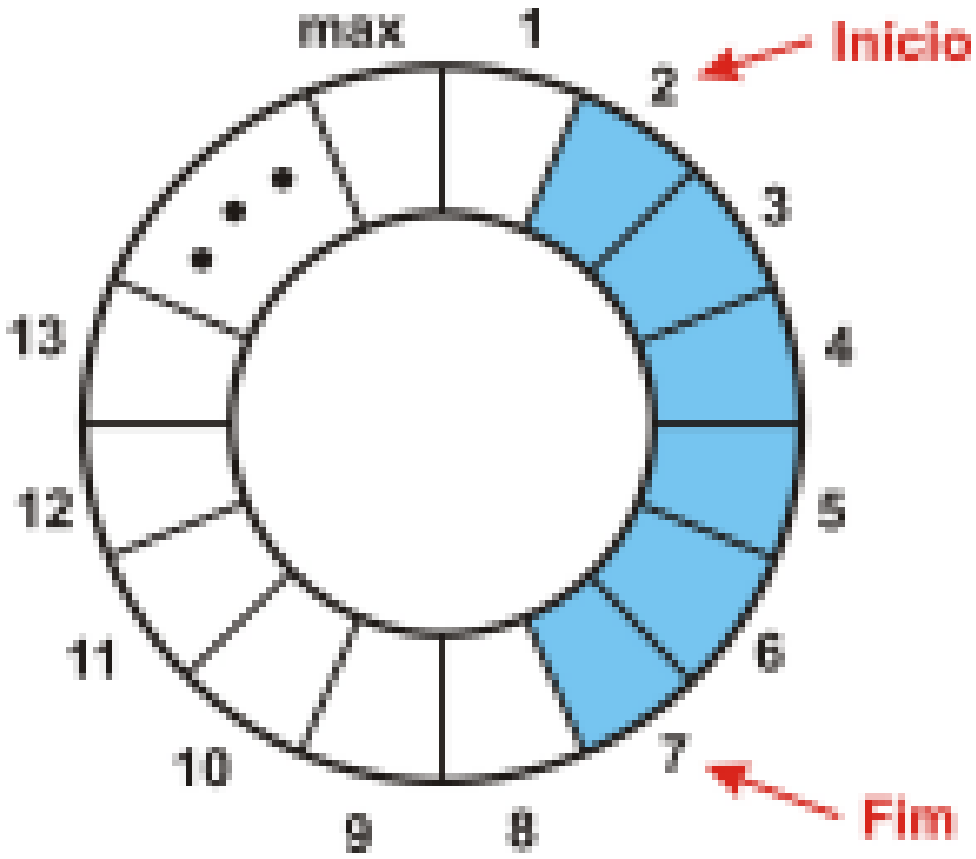
- Os itens são armazenados em posições contíguas de memória.
- A operação Enfileira faz a parte de trás da fila expandir-se.
- A operação Desenfileira faz a parte da frente da fila contrair-se.
- A fila tende a caminhar pela memória do computador, ocupando espaço na parte de trás e descartando espaço na parte da frente.

# Filas

TAD Fila: Implementação por ARRAY

- Com poucas inserções e retiradas, a fila vai ao encontro do limite do espaço da memória alocado para ela.

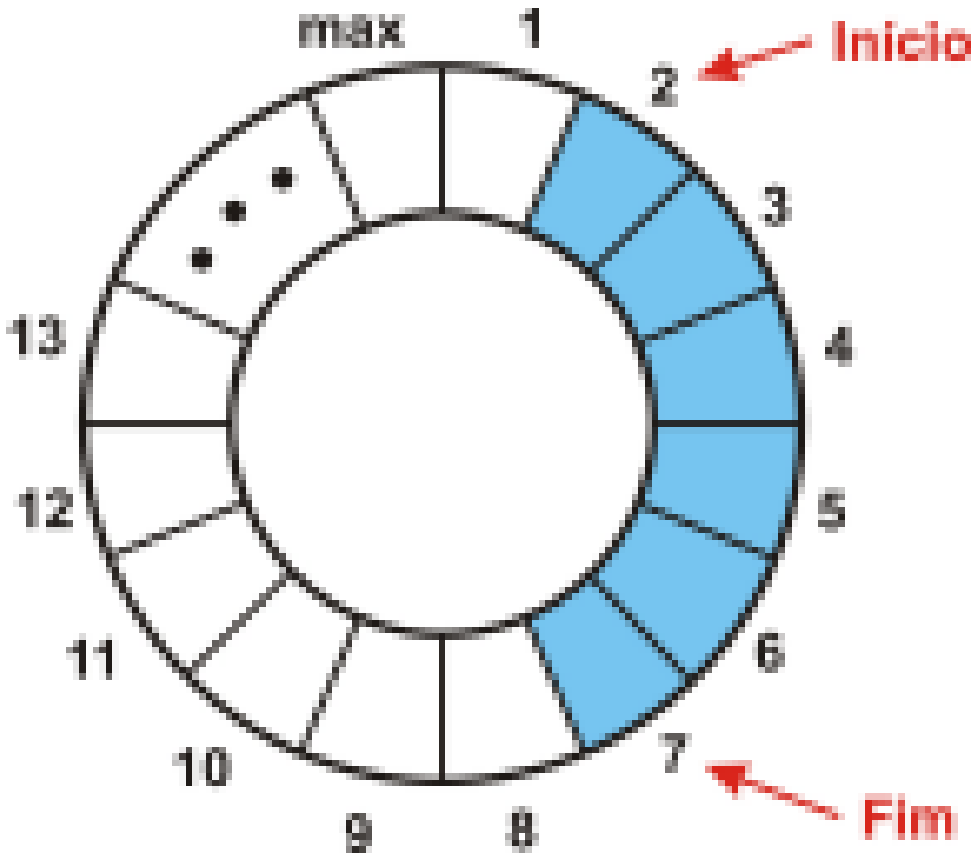
- Solução: imaginar o array como um círculo. A primeira posição segue a última.



# Filas

TAD Fila: Implementação por ARRAY

- A fila se encontra em posições contíguas de memória, em alguma posição do círculo, delimitada pelos apontadores Início e Fim.
- Para enfileirar, basta mover o apontador Fim uma posição no sentido horário.
- Para desenfileirar, basta mover o apontador Início uma posição no sentido horário.



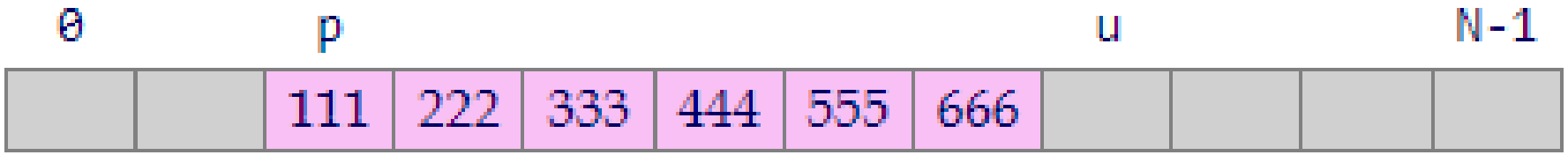
# Filas

TAD Fila: Implementação por ARRAY

- Suponha que nossa fila mora em um vetor fila[0..N-1]. (A natureza dos elementos do vetor é irrelevante: eles podem ser inteiros, bytes, ponteiros, etc.)

- Digamos que a parte do vetor ocupada pela fila é fila[p..u-1].

- O primeiro elemento da fila está na posição p e o último na posição u-1. A fila está vazia se p == u e cheia se u == N.



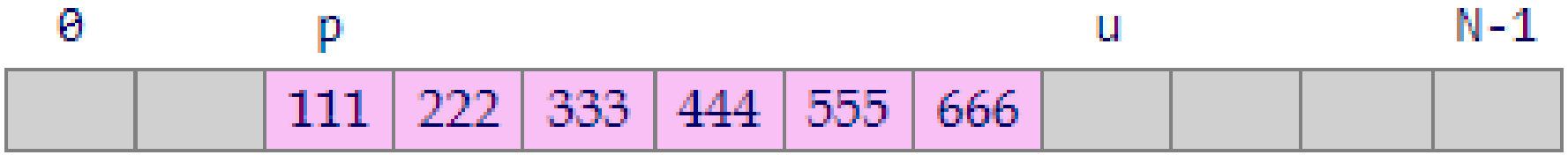
# Filas

TAD Fila: Implementação por ARRAY

- Para tirar, ou remover, um elemento da fila basta fazer

```
x = fila[p++];
```

- Isso equivale ao par de instruções  x = fila[p]; p += 1;,  nesta ordem.
- É claro que você só deve fazer isso se tiver certeza de que a fila não está vazia.



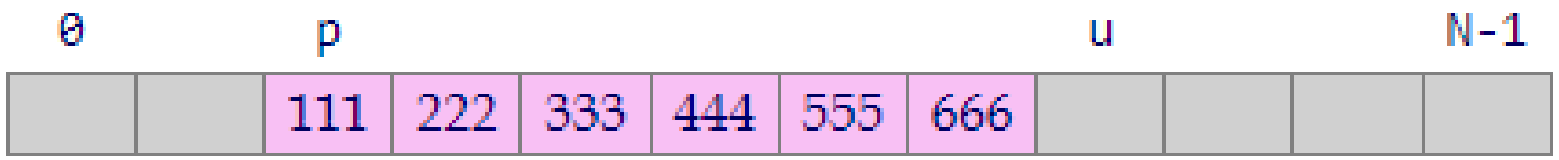
# Filas

TAD Fila: Implementação por ARRAY

- Para colocar, ou inserir, um objeto y na fila basta fazer
  fila[u++] = y;
- Isso equivale ao par de instruções  fila[u] = y; u += 1;,  nesta ordem.

```
int tiradafila (void) {
return fila[p++];
}

void colocanafila (int y) {
fila[u++] = y;
}
```



# Filas

TAD Fila: Implementação por ARRAY

```c
#define N 100
int fila[N],int p, u;
int dist[N];

void criafila (void) {
p = u = 0;
}

int filavazia (void) {
return p >= u;
}
```

```c
int tiradafila (void) {
return fila[p++];
}

void colocanafila (int y) {
fila[u++] = y;
}
```

# Filas

TAD Fila: Implementação por PONTEIRO

- A fila é implementada por meio de células.
- Cada célula contém um item da fila e um apontador para outra célula.
- Há uma célula cabeça para facilitar a implementação das operações Enfileira e Desenfileira quando a fila está vazia.

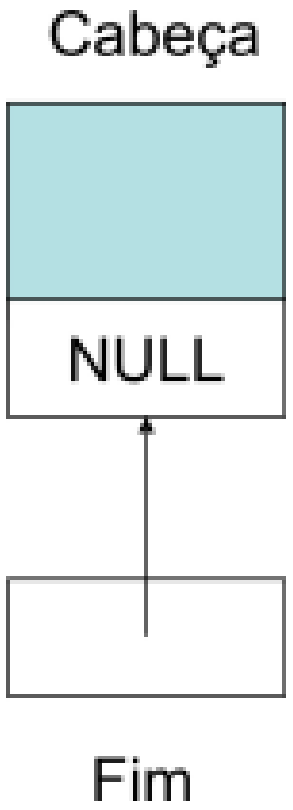
# Filas

TAD Fila: Implementação por PONTEIRO

- A fila contém um apontador para o início da fila (célula Cabeça) e um apontador para a parte de trás da fila (Fim).
- Quando a fila está vazia, os apontadores Cabeça e Fim apontam para a célula cabeça.
- Para enfileirar um novo item, basta criar uma célula nova, ligá-la após a célula que contém xn e colocar nela o novo item.
- Para desenfileirar o item x1, basta desligar a célula após a cabeça da lista

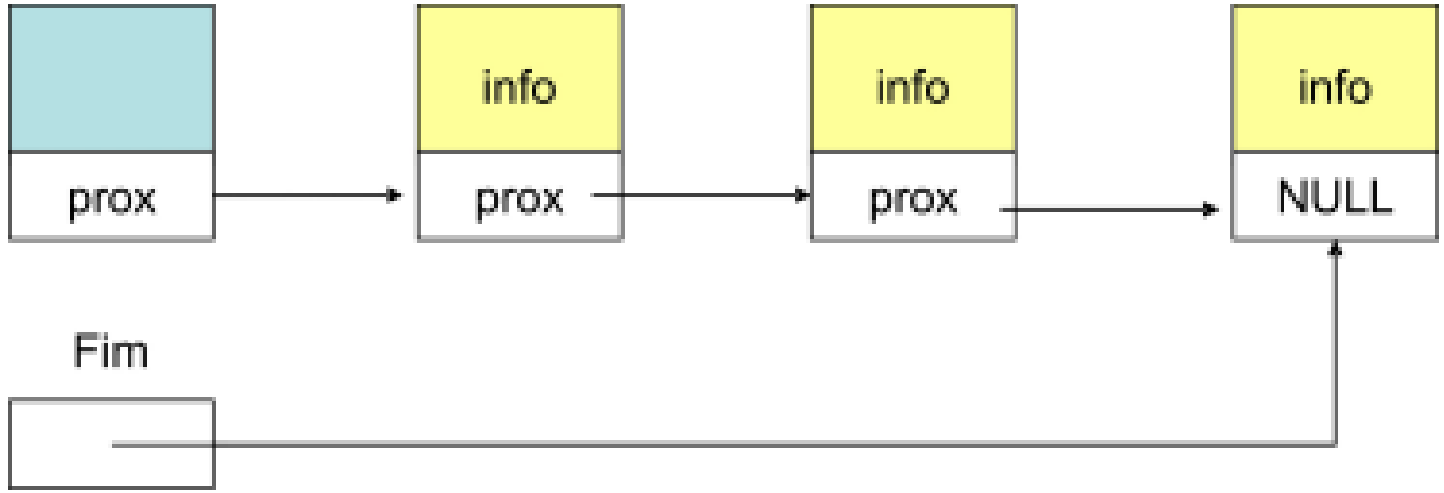
# Filas

TAD Fila: Criar Fila Vazia (usando célula Cabeça)



# Filas

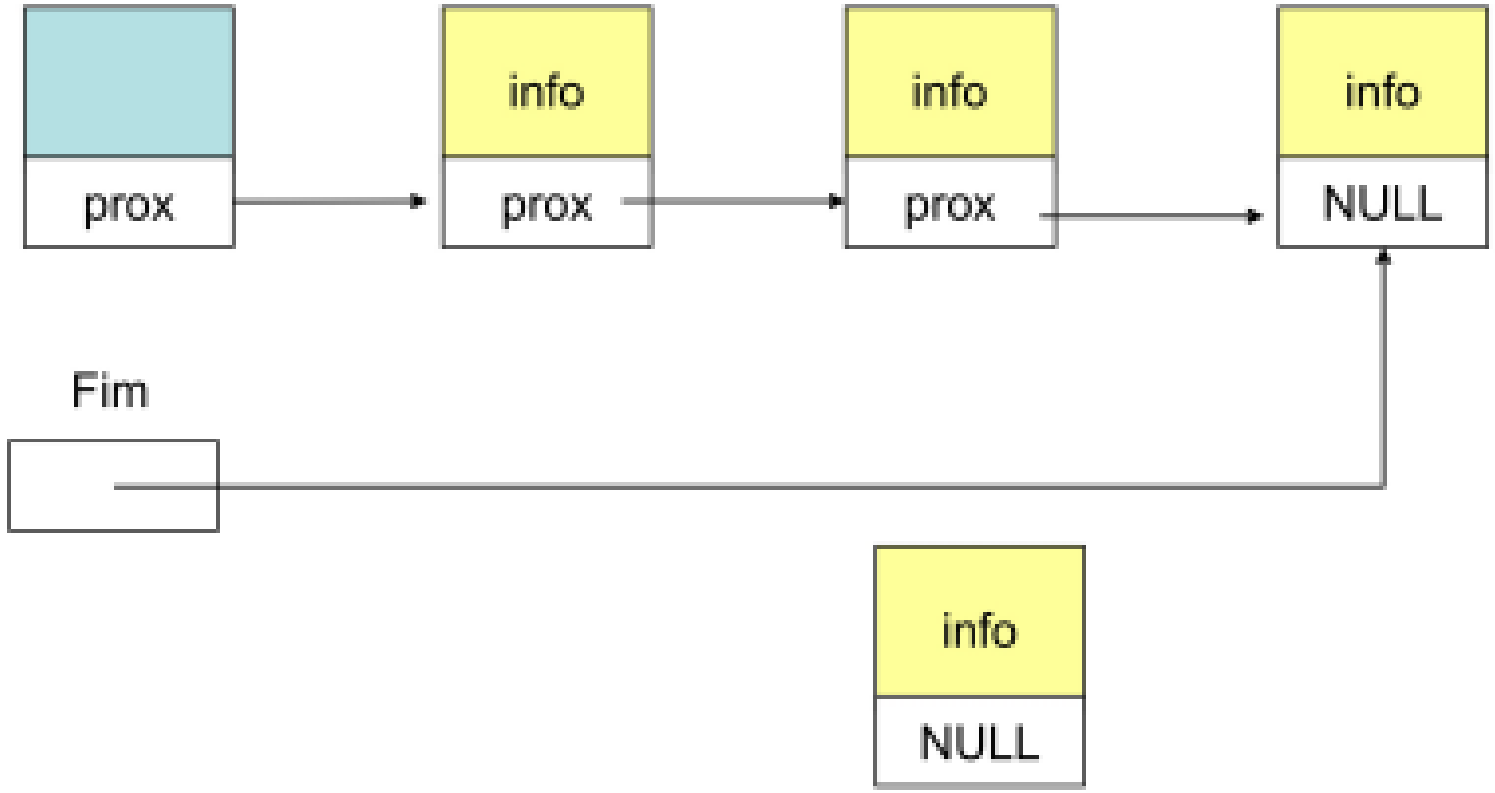
TAD Fila: INSERÇÃO de Novos Elementos




Opção única de posição onde se pode inserir: Final da filaa, ou seja, última posição.

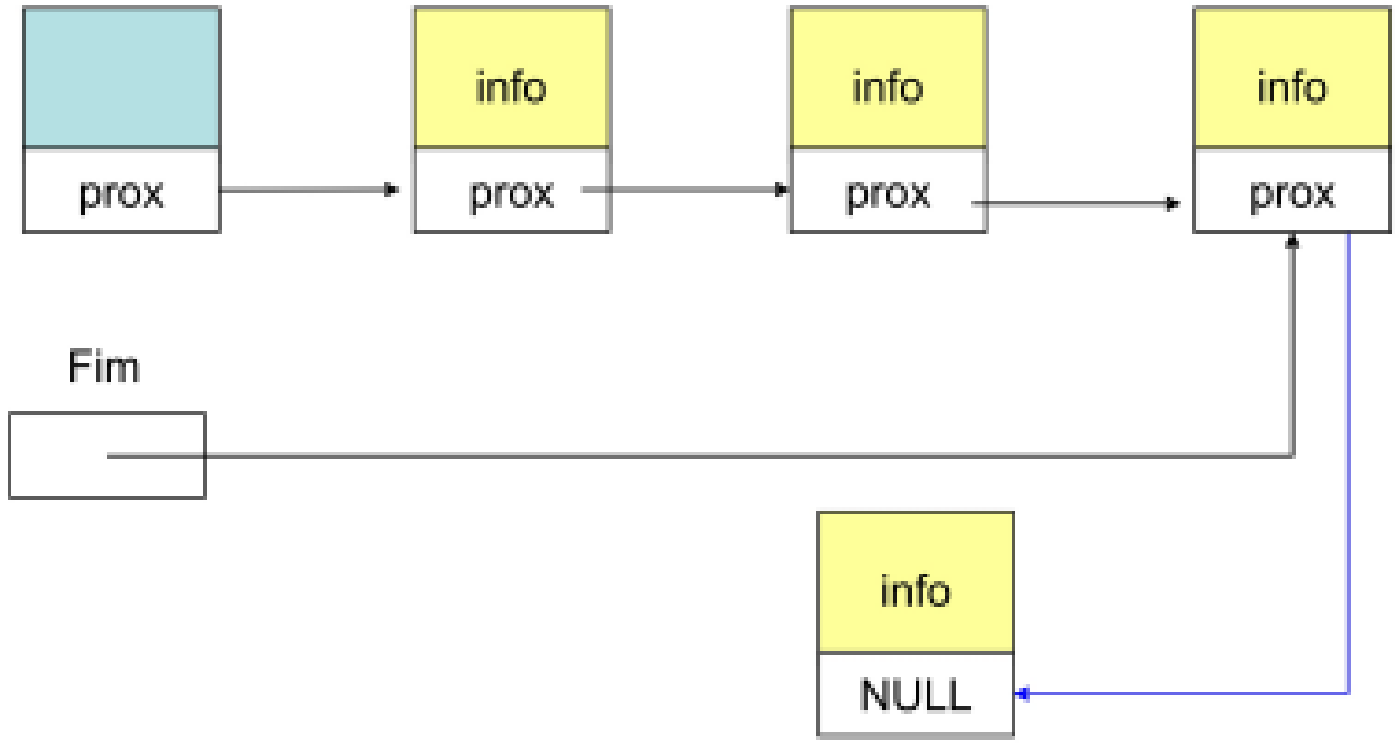
# Filas

TAD Fila: INSERÇÃO de Novos Elementos



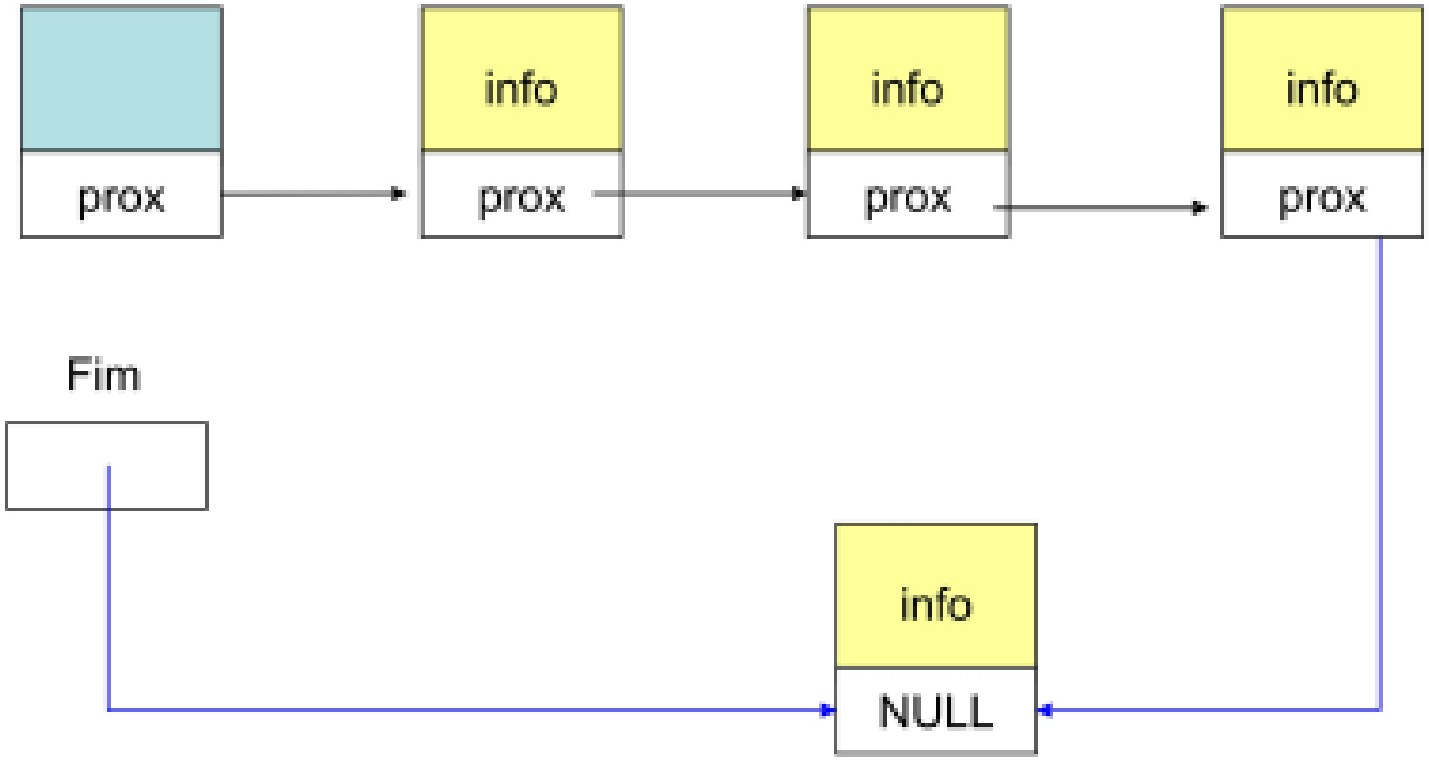
# Filas

TAD Fila: INSERÇÃO de Novos Elementos



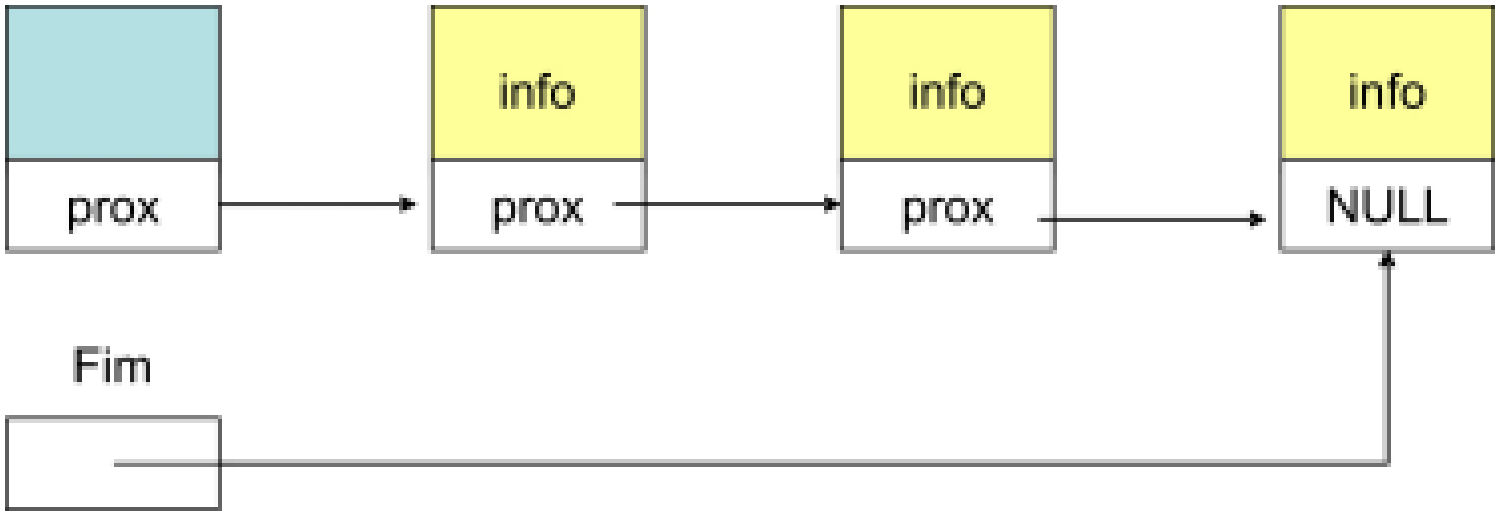
# Filas

TAD Fila: INSERÇÃO de Novos Elementos



# Filas

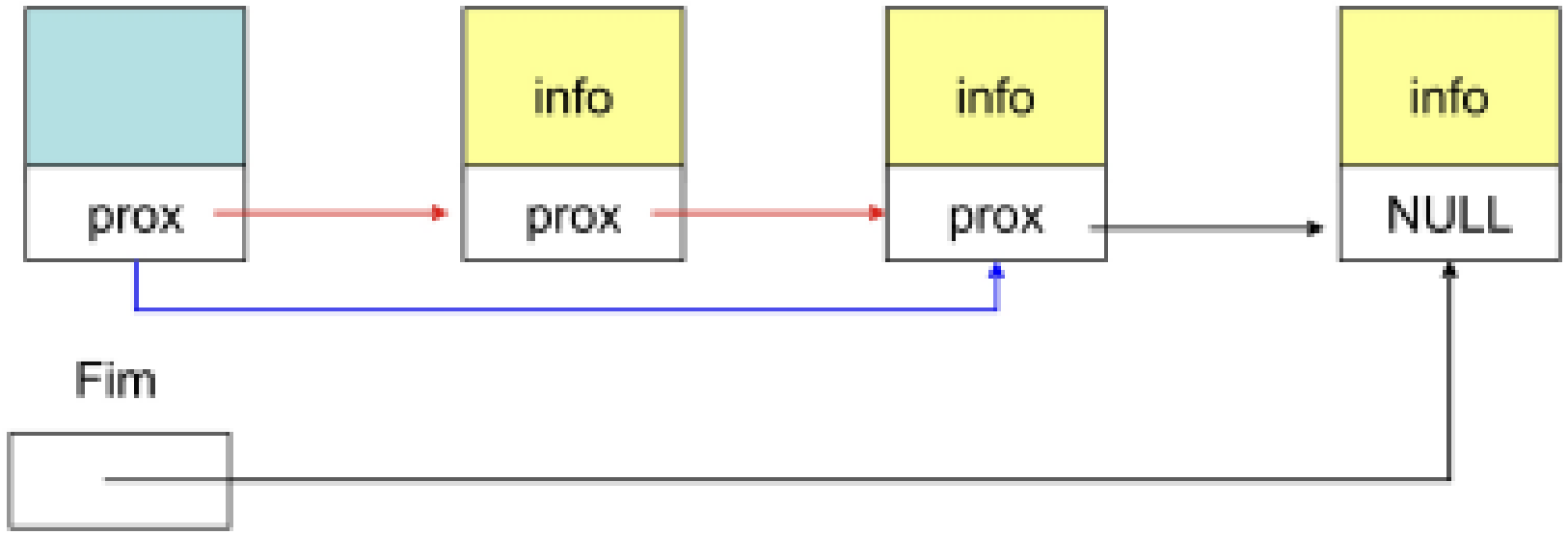
TAD Fila: RETIRADA de Elementos




Opção única de posição onde se pode retirar: Início da fila, ou seja, primeira posição.

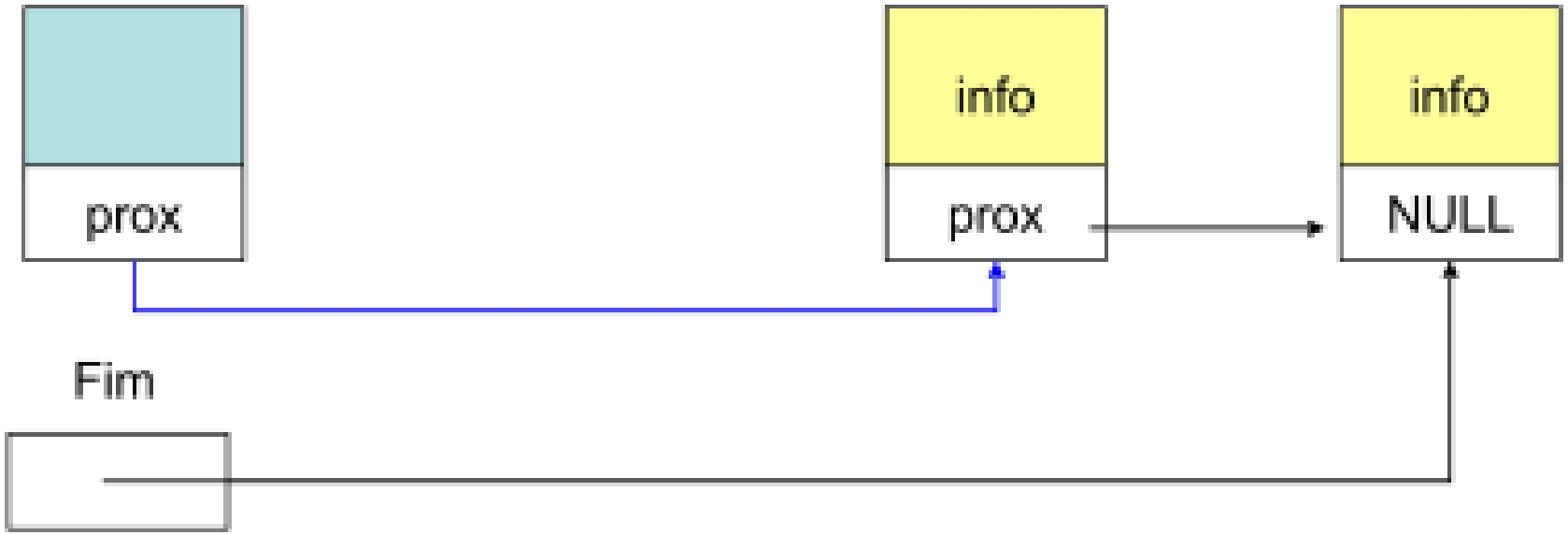
# Filas

TAD Fila: RETIRADA de Elementos



# Filas

TAD Fila: RETIRADA de Elementos



# Filas

## Implementação de Filas com Lista

- O nó da lista para armazenar valores reais, pode ser dado por:

```
struct no {
    float info;
    struct no* prox;
};
typedef struct no No;
```

- A estrutura da fila agrupa os ponteiros para o início e o fim da lista:

```
struct fila {
    No* ini;
    No* fim;
};
```

# Filas

Implementação de Filas com Lista

- Para criar uma fila, alocamos o espaço para a estrutura, e inicializamos seus dois ponteiros para NULL.

```
fila_t * fila_cria(void){
    /*Inicializa nosso novo objeto do tipo fila_t *  */
    fila_t * nova_fila = malloc(sizeof(fila_t));
    nova_fila -> frente = NULL;
    nova_fila -> final = NULL;
    return nova_fila;
}
```

# Filas

Implementação de Filas com Lista

- Agora vamos implementar a nossa função de Inserir. Ela deve pegar um valor e inseri-lo ao final da fila:

```
void fila_put(fila_t * fila, int val){
    /*Cria o novo nó*/
    fila_no * novo_no = malloc(sizeof(fila_no));
    novo_no -> dado = val;
    novo_no -> prox = NULL;
    /*Se a fila estiver vazia,
    a frente e o final apontam para o mesmo novo elemento*/
    if (fila -> frente == NULL){
        fila -> frente = fila -> final = novo_no;
    }
```

# Filas

Implementação de Filas com Lista

- Agora vamos implementar a nossa função de Inserir. Ela deve pegar um valor e inseri-lo ao final da fila:

```
        /*Caso contrário...*/
            else{
            /*O que era o último elemento, vai
            apontar para o novo último elemento*/
            fila -> final -> prox = novo_no;
                    /*O novo elemento vai ser o final da fila*/
            fila -> final = novo_no;
        }
}
```

# Filas

Implementação de Filas com Lista

- Agora vamos implementar a nossa função de Remover.
- Ela deve pegar o primeiro elemento da fila.

```
int fila_get(fila_t * fila){
    /*Prossiga apenas se realmente tiver objetos na fila*/
    if (fila -> frente){
        /*Armazena o dado do objeto a ser removido*/
        int val = fila -> frente -> dado;
        /*Precisamos armazenar o endereço do primeiro elemento
        porque vamos mudar o ponteiro fila -> frente, mas
        precisamos liberar o primeio elemento com free() */
        fila_no * temp = fila -> frente;
        /*A fila anda...*/
        fila -> frente = fila -> frente -> prox;
        free(temp);
        return val;
    }
    /*Avise se o ponteiro for NULL*/
    return -1;
}
```

# Filas

Implementação de Filas com Lista

- Imprimindo a fila.

```
void fila_imprime(fila_t * fila){
    /*Se a fila existir...*/
    if (fila){
        fila_no * aux = fila -> frente;
        while(aux){
            printf("[%d]->", aux -> dado);
            aux = aux -> prox;
        }
    }
    printf("NULL\n");
}
```

# FIM!