



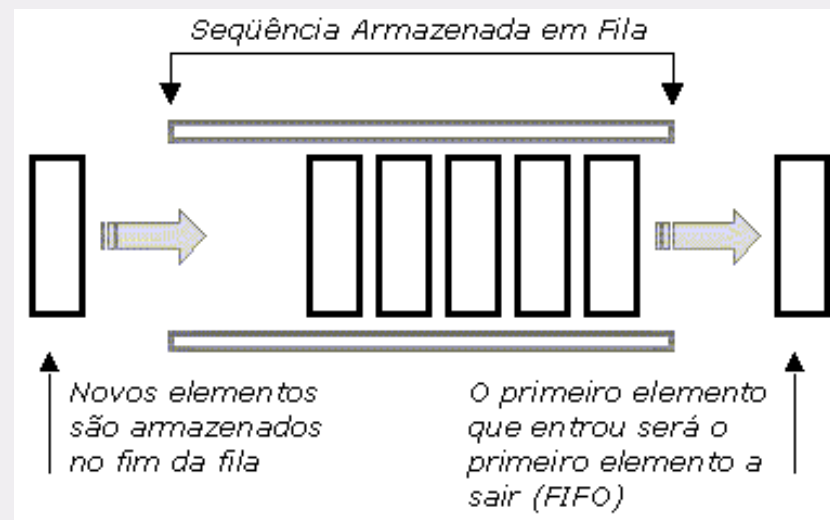
Agenda

Unidade 3: Listas Lineares

1. Listas lineares
2. Pilhas
- 3. Filas**
4. Implementações

Filas

- As filas (*queues*) são conjuntos de elementos (ou listas) cujas operações de inserção são feitas por uma extremidade e as de remoção, por outra extremidade.
- Analogia: fila de espera em que as pessoas no início da fila são servidas primeiro e as pessoas que chegam entram no fim da fila.
- São chamadas listas FIFO (“first-in”, “first-out”).






TAD Filas

■ Conjunto de operações:

1. `FFVazia(Fila)`. Faz a fila ficar vazia.
2. `Enfileira(x, Fila)`. Insere o item `x` no final da fila.
3. `Desenfileira(Fila, x)`. Retorna o item `x` no início da fila, retirando-o da fila.
4. `Vazia(Fila)`. Esta função retorna `true` se a fila está vazia; senão retorna `false`.

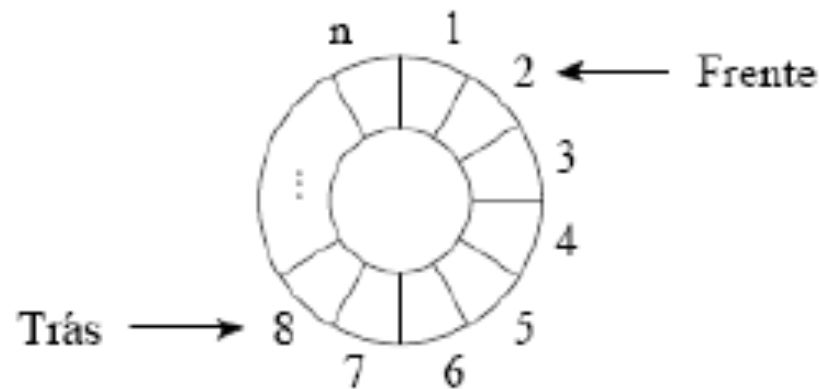


Implementação de Filas por meio de arranjos

- Os itens são armazenados em posições contíguas de memória
- Enfileirar (Enqueue): faz a **parte de trás** da fila expandir-se.
- Desenfileirar (Dequeue): 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.

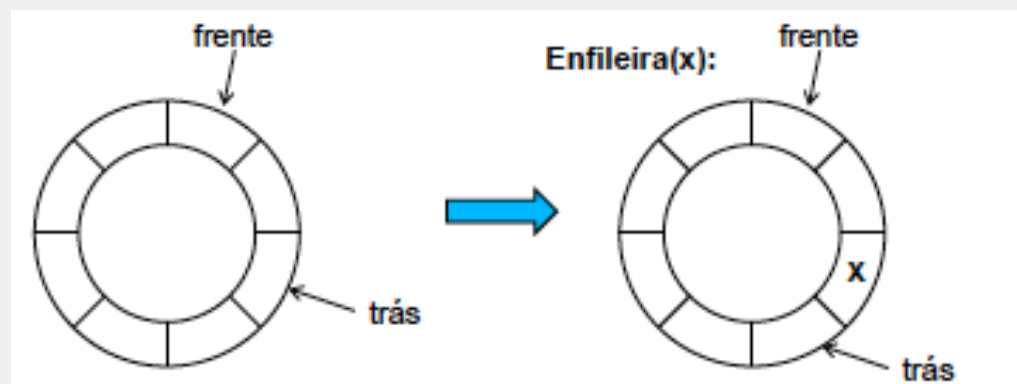
Implementação de Filas por meio de arranjos

- 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.



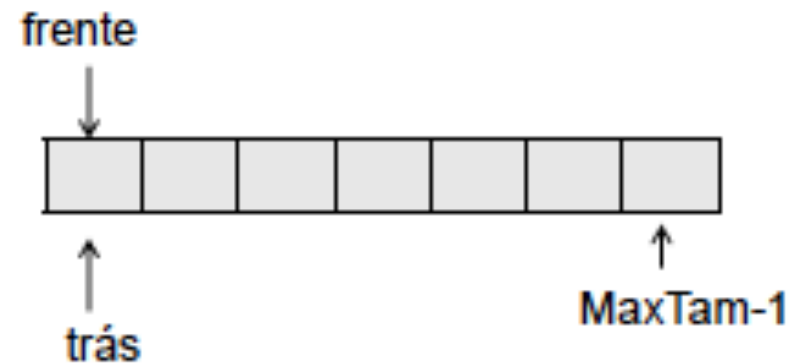
Implementação de Filas por meio de arranjos

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



Estrutura da Fila Usando Arranjo

```
#define MAXTAM 1000
typedef int TipoApontador;
typedef int TipoChave;
typedef struct {
    TipoChave Chave;
    /* outros componentes */
} TipoItem;
typedef struct {
    TipoItem Item[MAXTAM];
    TipoApontador Frente, Tras;
} TipoFila;
```



Operações sobre filas Usando Arranjos

- Nos casos de fila cheia e fila vazia, os apontadores Frente e Trás apontam para a mesma posição do círculo.

```
void FFVazia(TipoFila *Fila)
{
    Fila->Frente = 0;
    Fila->Tras = Fila->Frente;
} /* FFVazia */

int Vazia(TipoFila *Fila)
{
    return (Fila->Frente == Fila->Tras);
} /* Vazia */
```

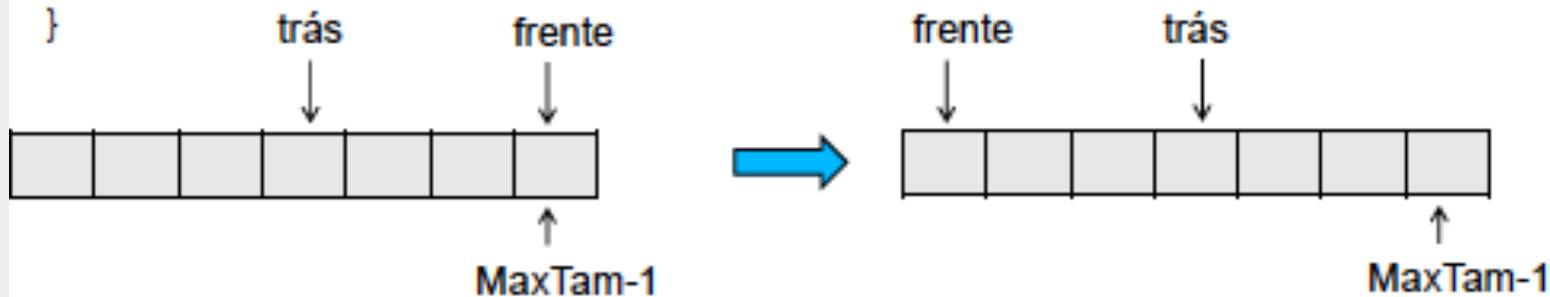

Operações sobre filas Usando Arranjos

```
int Enfileira(TipoItem x, TipoFila *Fila) {  
    if ((Fila->Tras + 1) % MaxTam == Fila->Frente){  
        printf("Erro: fila está cheia\n"); return 0;  
    }  
    else {  
        Fila->Item[Fila->Tras] = x;  
        Fila->Tras = (Fila->Tras + 1) % MaxTam;  
    }  
    return 1;  
}
```



Operações sobre filas Usando Arranjos

```
TipoItem Desenfileira(TipoFila *Fila) {  
    if (Vazia(Fila)) {  
        printf("Erro: fila está vazia\n"); ERRO;  
    }  
    else {  
        int idx = Fila->Frente;  
        Fila->Frente = (Fila->Frente + 1) % MaxTam;  
        return Fila->Item[idx];  
    }  
}
```



Implementação de Filas por meio de Apontadores

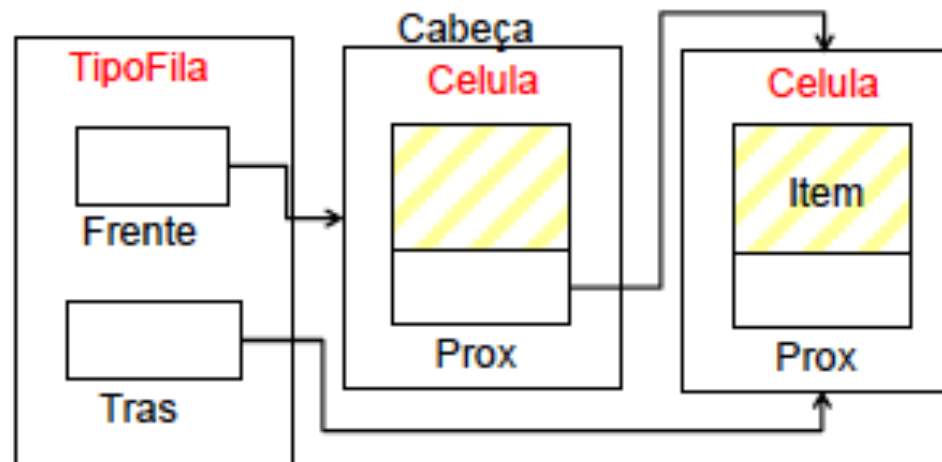
- Utiliza célula cabeça para facilitar a implementação das operações Enfileira e Desenfileira quando a fila está vazia.
- **Quando vazia:** apontadores Frente e Trás apontam para a célula cabeça.
- **Enfileirar novo item:** criar uma célula nova, ligá-la após a célula contendo x_n e colocar nela o novo item.
- **Desenfileirar:** desligar a célula cabeça da lista e a célula que contém x_1 passa a ser a célula cabeça.



Implementação de Filas por meio de Apontadores

```
typedef int TipoChave;  
  
typedef struct TipoItem {  
    TipoChave Chave;  
    /* outros componentes */  
} TipoItem;  
  
typedef struct Celula *Apontador;  
  
typedef struct Celula {  
    TipoItem Item;  
    Apontador Prox;  
} Celula;  
  
typedef struct TipoFila {  
    Apontador Frente, Tras;  
} TipoFila;
```

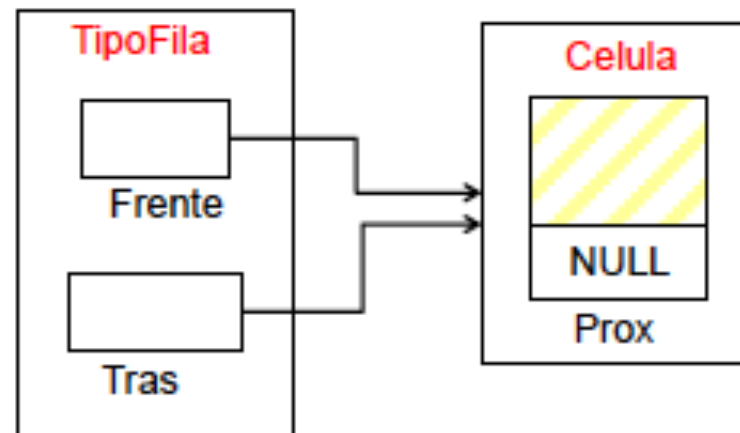
- A fila é implementada por meio de células.
- Cada célula contém um item da fila e um apontador para outra célula.
- A estrutura TipoFila contém um apontador para a frente da fila (célula cabeça) e um apontador para a parte de trás da fila.



Operações sobre Filas Usando Apontadores

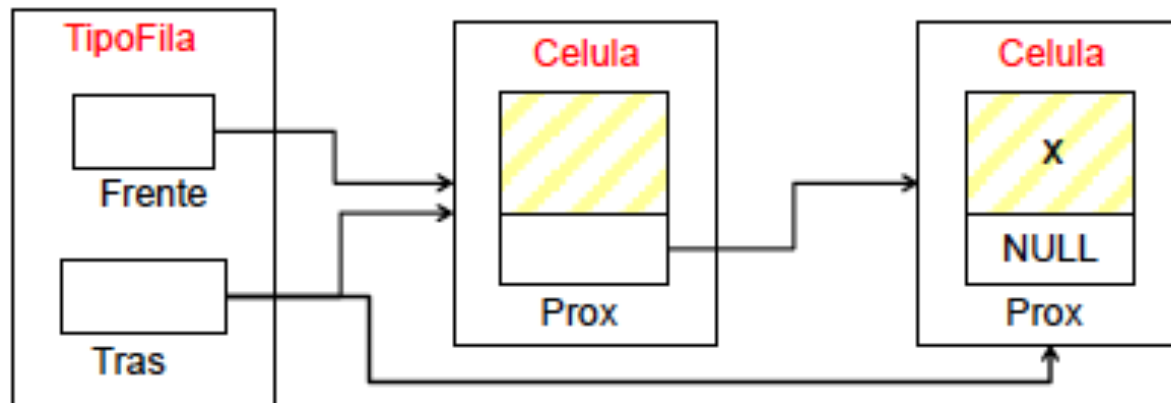
```
void FFVazia(TipoFila *Fila)
{
    Fila->Frente = (Apontador) malloc(sizeof(Celula));
    Fila->Tras = Fila->Frente;
    Fila->Frente->Prox = NULL;
} /* FFVazia */
```

```
int Vazia(TipoFila *Fila)
{
    return (Fila->Frente == Fila->Tras);
} /* Vazia */
```



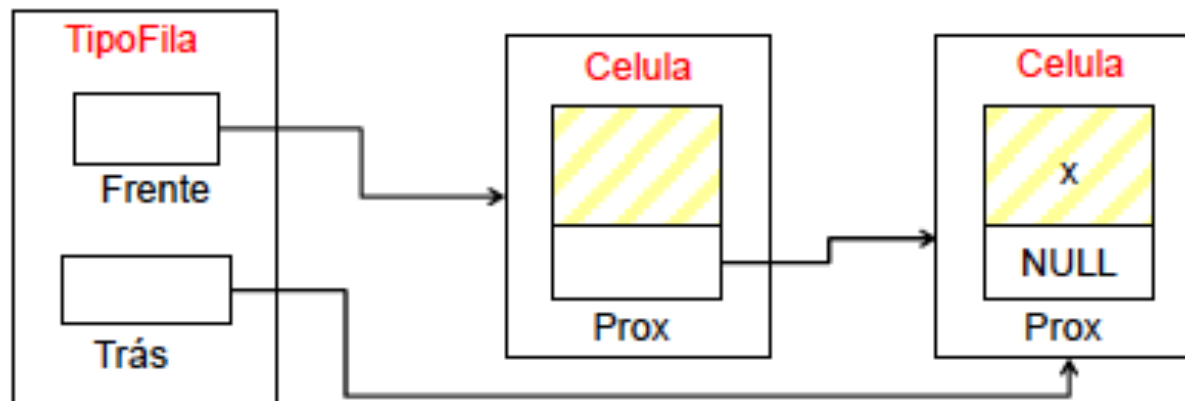
Operações sobre Filas Usando Apontadores

```
void Enfileira(TipoItem x, TipoFila *Fila)
{
    Fila->Tras->Prox = (Apontador) malloc(sizeof(Celula));
    Fila->Tras = Fila->Tras->Prox;
    Fila->Tras->Item = x;
    Fila->Tras->Prox = NULL;
}
```



Operações sobre Filas Usando Apontadores

```
TipoItem Desenfileira(TipoFila *Fila){  
    Apontador q;  
    if (Vazia(Fila)) {  
        printf("Erro: fila está vazia\n"); ERRO;  
    }  
    q = Fila->Frente;  
    Fila->Frente = Fila->Frente->Prox;  
    free(q);  
    return Fila->Frente->Item;  
}
```





Exercício de Fixação

- Implemente uma fila usando arranjos
- Escreva uma função que devolva o comprimento (ou seja, o número de elementos) da fila.



Referências

- Nivio Ziviani. Projeto de Algoritmos com Implementações em Pascal e C