

Estruturas de Dados

Filas





Conceito

- Coleção ordenada de itens (lista ordenada) em que a inserção de um novo item se dá em um dos lados – no fim – e a remoção no outro lado – no início.
 - Listas FIFO/LILO (**F**irst **I**n **F**irst **O**ut/ **L**ast **I**n **L**ast **O**ut).
- Modelos intuitivos de filas são as linhas para comprar bilhetes de cinema e de caixa de supermercado.
- A fila, como a pilha, é conceitualmente uma estrutura dinâmica que está continuamente mudando pois itens são adicionados/retirados.

TAD – Fila – Operações

//Cria uma pilha vazia. Deve ser usado antes de qualquer outra operação

```
void definir(fila *f);
```

//Reinicializa uma fila existente q como uma fila vazia. Dependendo da
//implementação da estrutura de dados deve remover todos os seus elementos.

```
void tornar_vazia(fila *f);
```

//Retorna true (1) se fila não contém elementos, false (0) caso contrário.

```
int vazia(fila *f);
```

//Adiciona um item no fim da fila q. Retorna true (1) se operação realizada
//com sucesso, false (0) caso contrário.

```
int inserir(fila *f, tipo_elem item);
```

//Remove um item do início da fila q. Retorna true (1) se operação
//realizada com sucesso, false (0) caso contrário.

```
int remover(fila *f, tipo_elem item);
```

//Retorna o tamanho da fila.

```
int tamanho(fila *f);
```

//Mostra o começo da fila sem remover o item. Retorna true (1) se operação
//realizada com sucesso, false (0) caso contrário.

```
int comeco_fila (fila *f, tipo_elem *item);
```



Implementações de Filas: Estática

Há um meio de se utilizar de um vetor na implementação de uma fila?

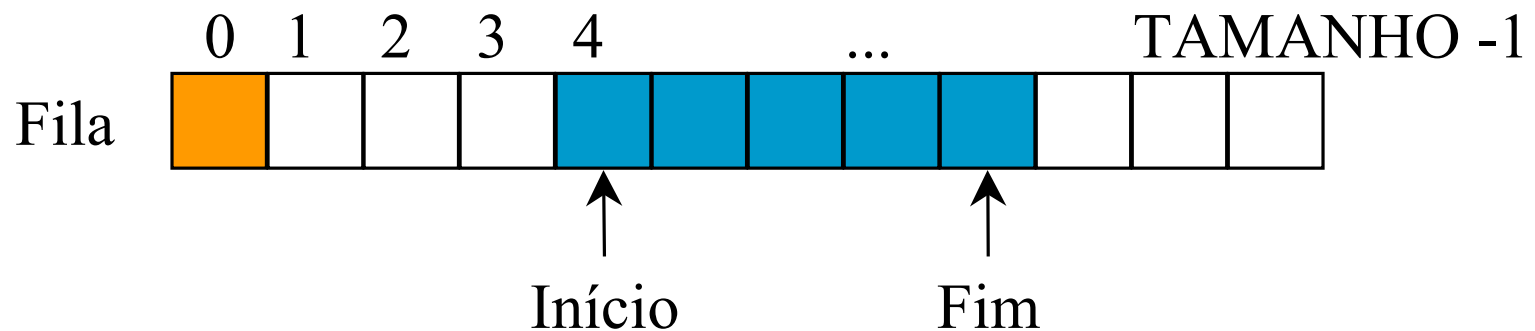
SIM se nós dimensionarmos o vetor com um tamanho que dê para acomodar o tamanho máximo da fila, e além disso precisamos dos ponteiros FIM e COMEÇO.

Implementações de Filas: Estática

```
#define TAMANHO 100
```

```
typedef int tipo_elem;
```

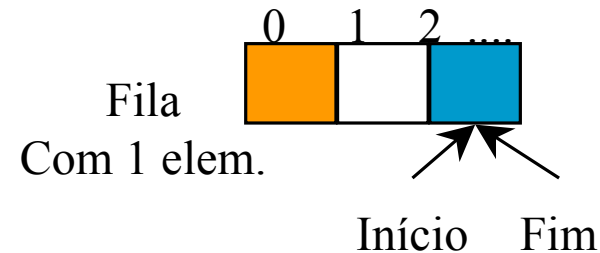
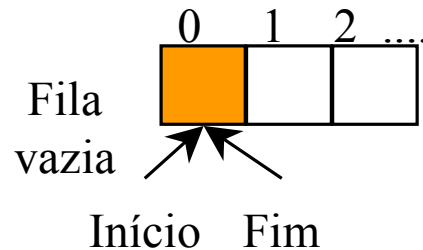
```
typedef struct _fila fila;  
struct _fila{  
    int inicio, fim;  
    tipo_elem vetor[TAMANHO];  
};
```



Implementações de Filas: Estática

O que é então uma fila vazia?

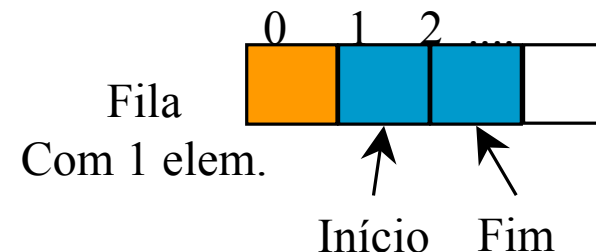
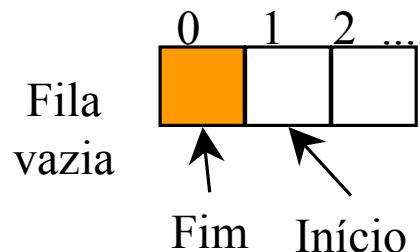
Solução 1: início = fim



Mas quando o primeiro item é inserido os ponteiros início e fim irão ter o mesmo valor, pois o item está no início e fim da fila. Portanto, não conseguimos diferenciar fila vazia de fila com um elemento.

Solução 2: Fim = 0 e Início = 1

- Fila vazia = $\text{fim} < \text{início}$
- 0 (zero) não é mais uma posição válida para alocar elementos.



Implementações de Filas: Estática

Inserir

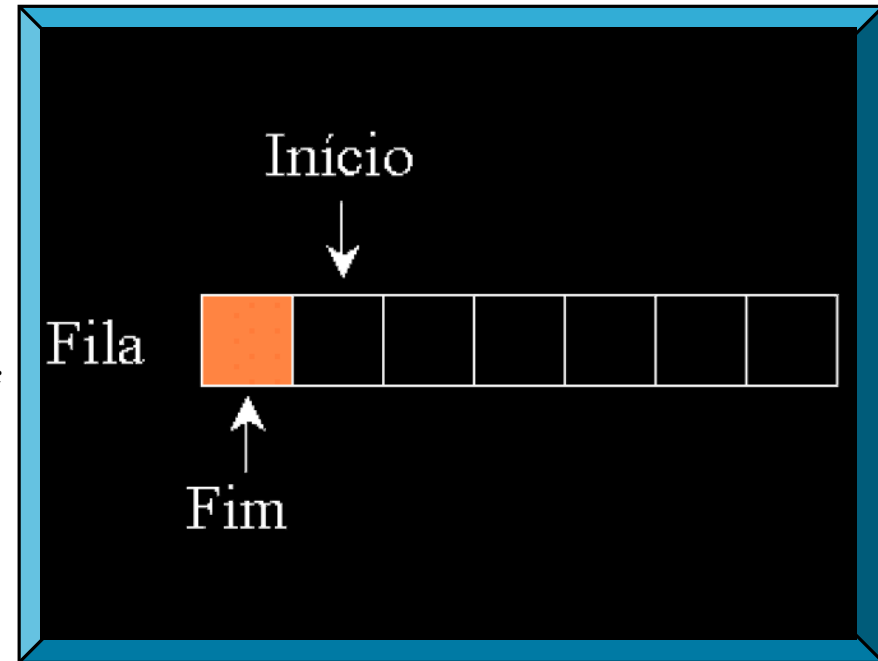
```
f->fim = f->fim+1;  
f->vetor[f->fim] = item;
```

Remover

```
f->inicio = f->inicio+1;  
item = f->vetor[f->inicio];
```

Tamanho

```
f->fim - f->inicio+1;
```



Problema:

E se quisermos inserir um novo elemento? Há espaço à esquerda MAS nossas operações não “vêem” isso.

Solução 1: Modificar remover: cada remoção desloca os elementos de 1 posição à esquerda. E se a Fila = 5000? Ineficiente.

Solução 2: Visualizar a fila como um vetor circular.

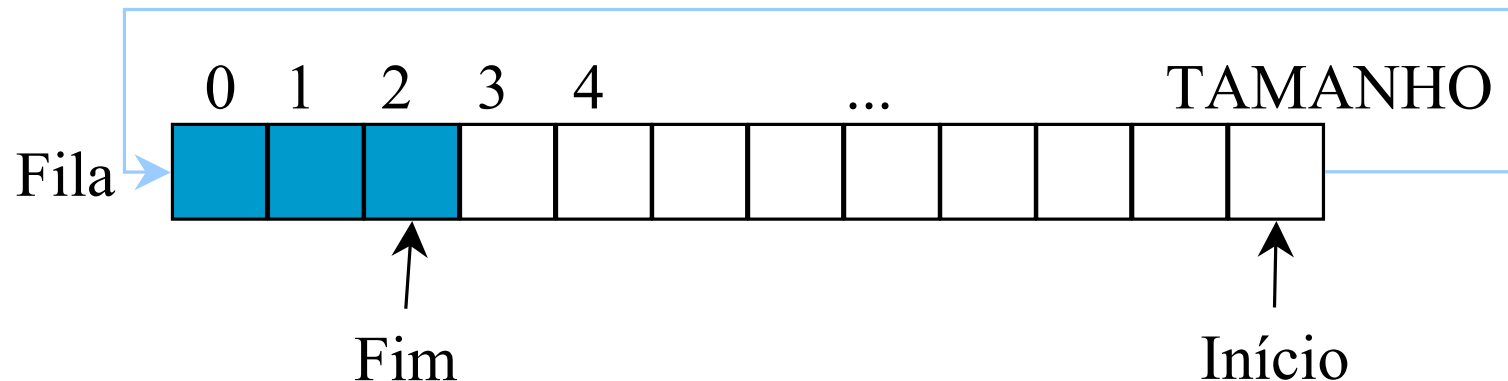
Fila vazia: fim == inicio

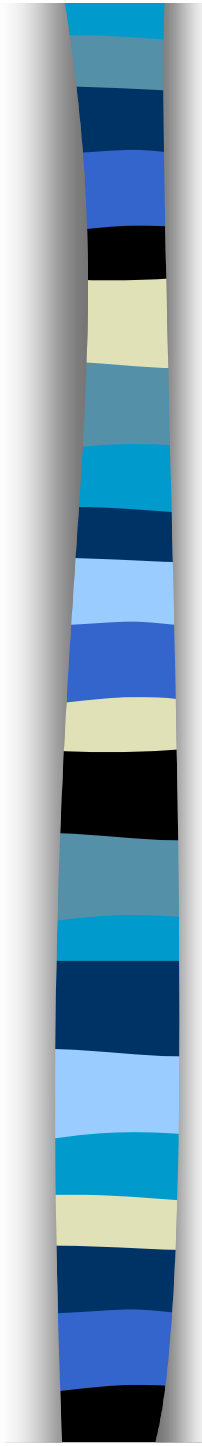
Implementações de Filas: Circular Estática

```
#define TAMANHO 100 //Numero maximo de itens na fila  
// Existe um Espaco em  
//branco para diferenciar fila cheia de fila vazia
```

```
typedef int tipo_elem;
```

```
typedef struct _fila fila;  
struct _fila{  
    int inicio, fim;  
    tipo_elem vetor[TAMANHO];  
};
```





```
//Cria uma fila vazia. Deve ser usado antes de qualquer
//outra operação
void definir(fila *f){
    f->inicio = TAMANHO;
    f->fim = TAMANHO;
    //ponteiro atrasado; aponta para uma posição anterior
    //ao início
}

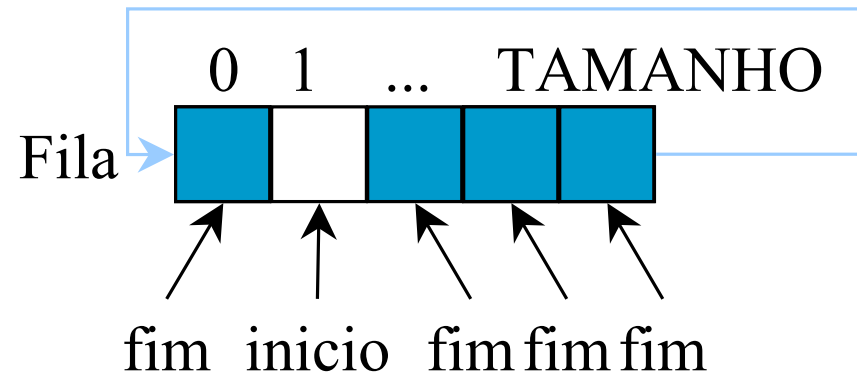
//Reinicializa uma fila existente q como uma fila vazia.
//Dependendo da implementação da estrutura de dados deve
//remover todos os seus elementos.
void tornar_vazia(fila *f){
    f->fim = TAMANHO;
    f->inicio = TAMANHO;
}

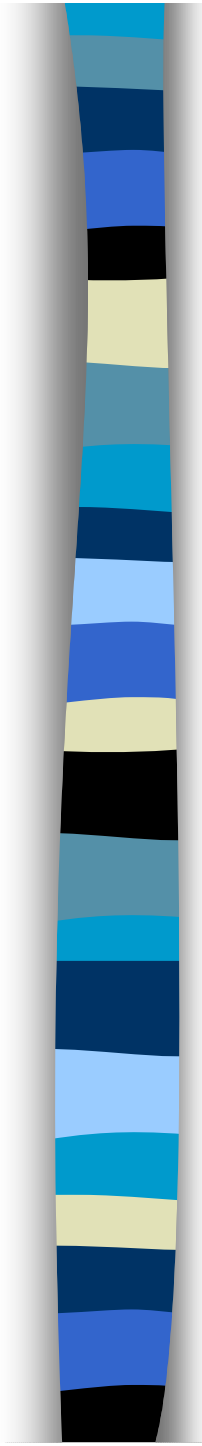
//Retorna true (1) se fila não contém elementos, false
//(0) caso contrário.
int vazia(fila *f){
    return (f->inicio == f->fim);
}
```

```
//Retorna true (1) se fila cheia, false (0) caso
//contrário.
int cheia(fila *f){
    //os dois ponteiros diferem de uma posição
    return (f->inicio == ((f->fim + 1) % TAMANHO)) ;
}
```

```
//Adiciona um item no fim da fila q. Retorna true (1) se
//operação realizada com sucesso, false (0) caso
//contrário.
```

```
int inserir(fila *f, tipo_elem item){
    //uma posição da fila nunca será preenchida
    if (!cheia(f)){
        f->fim = ((f->fim + 1) % TAMANHO);
        f->vetor[f->fim] = item;
        return 1;
    }
    return 0;
}
```



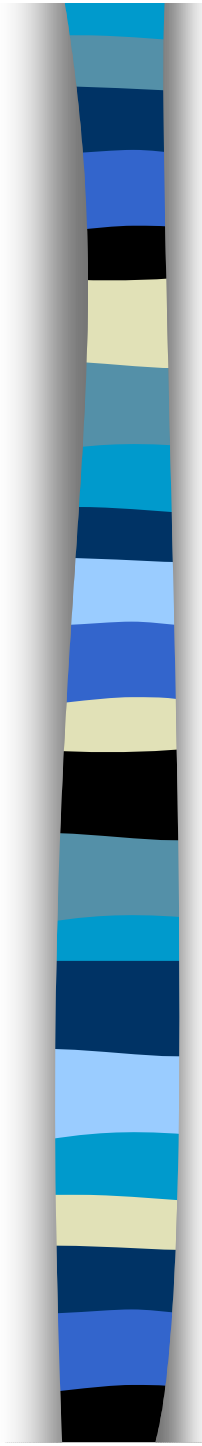


```
//Remove um item do início da fila q. Retorna 1 se  
//sucesso, 0 caso contrário.
```

```
int remover(fila *f, tipo_elem *item){  
    if (!vazia(f)){  
        f->inicio = (f->inicio + 1) % TAMANHO;  
        *item = f->vetor[f->inicio]; //opcional  
        return 1;  
    }  
    return 0;  
}
```

```
//Retorna o tamanho da fila.
```

```
int tamanho(fila *f){  
    if (vazia(f))  
        return 0;  
    else  
        if (f->inicio <= f->fim)  
            return (f->fim - (f->inicio + 1));  
        else  
            return (TAMANHO - f->inicio + f->fim + 1);  
}
```



```
//Mostra o começo da fila sem remove-lo. Retorna 1 se
//sucesso, 0 caso contrário.
int comeco_fila(fila *f, tipo_elem *item){
    if (!vazia(f)){
        *item = f->vetor[(f->inicio+1) %TAMANHO];
        return 1;
    }
    return 0;
}

//imprime a fila
void imprime(fila *f){
    int i = (f->inicio + 1) % TAMANHO;
    if (!vazia(f)){
        printf("\nFila: ");
        while (i != f->fim){
            printf("%d ", f->vetor[i]);
            i = (i + 1) % TAMANHO;
        }
    }else
        printf("\nFila vazia");
}
```

Implementações de Filas: Dinâmica

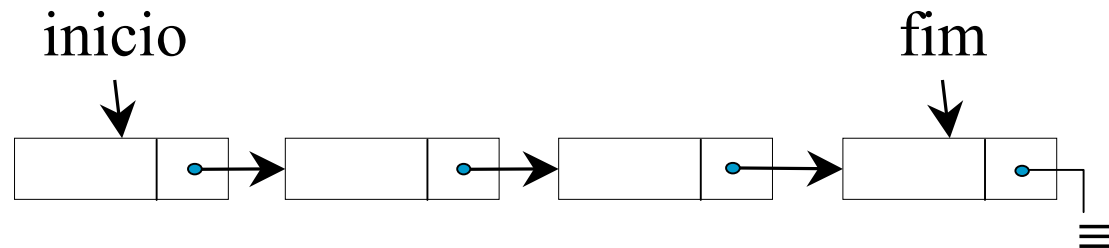
```
typedef int tipo_elem;
```

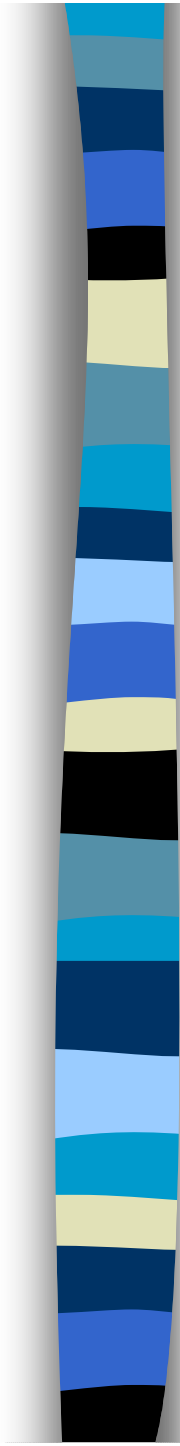
```
typedef struct _no no;
```

```
struct _no{  
    tipo_elem info;  
    no *proximo;  
};
```

```
typedef struct _fila fila;
```

```
struct _fila{  
    no *inicio, *fim;  
};
```





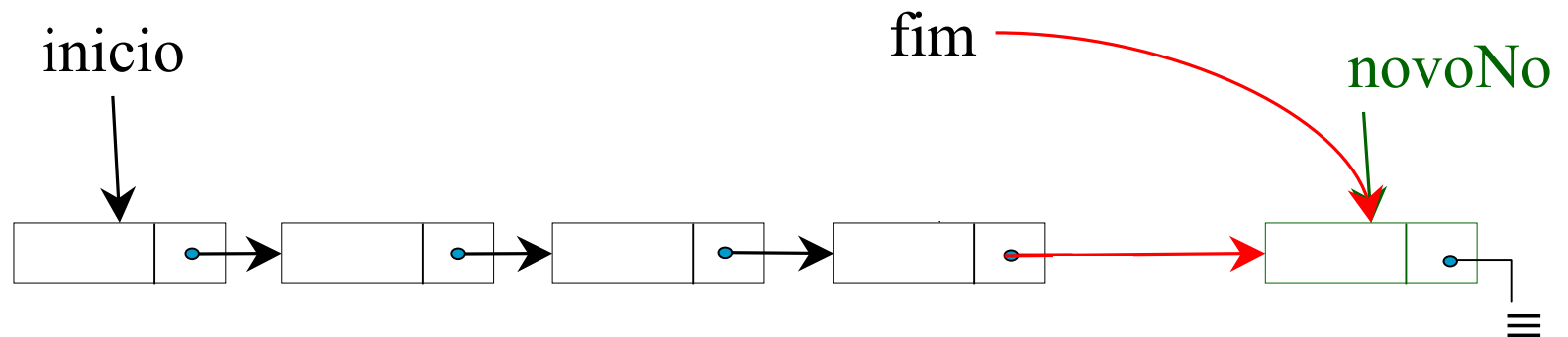
```
//Cria uma pilha vazia. Deve ser usado antes de qualquer outra
//operação
void definir(fila *f){
    f->inicio = NULL;
    f->fim = NULL;
}

//Reinicializa uma fila existente q como uma fila vazia.
//Dependendo da implementação da estrutura de dados deve
//remover todos os seus elementos.
void tornar_vazia(fila *f){
    no *tmp;
    if (!vazia(f)){
        tmp = f->inicio;
        while (f->inicio != NULL){
            f->inicio = (f->inicio)->proximo;
            free(tmp);
            tmp = f->inicio;
        }
    }
    f->fim = NULL;
}

//Retorna 1 se fila não contém elementos, 0 caso contrário.
int vazia(fila *f){
    return (f->inicio == NULL);
}
```

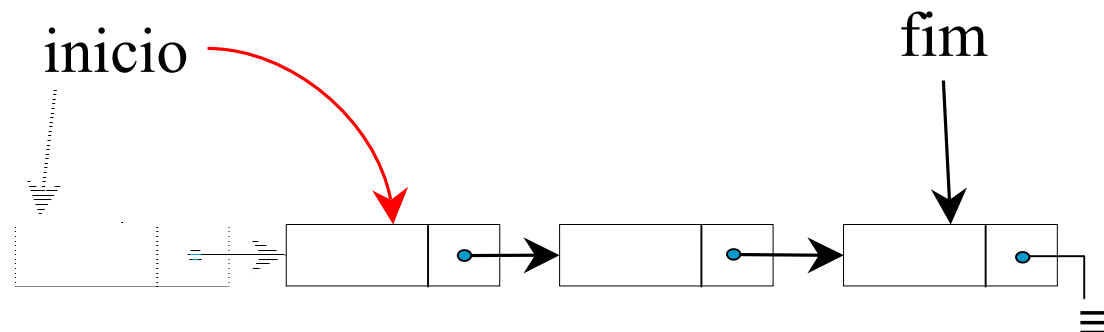
```
//Adiciona um item no fim da fila q. Retorna true (1) se  
//operação realizada com sucesso, false (0) caso contrário.
```

```
int inserir(fila *f, tipo_elem item){  
    no *novoNo = (no*)malloc(sizeof(no));  
    if (novoNo != NULL){  
        novoNo->info = item;  
        novoNo->proximo = NULL;  
        if (vazia(f))  
            f->inicio = novoNo; //Primeiro no  
        else  
            (f->fim)->proximo = novoNo;  
        f->fim = novoNo;  
        return 1;  
    }  
    return 0;  
}
```



```
//Remove um item do início da fila q. Retorna true (1) se  
//operação realizada com sucesso, false (0) caso  
//contrário.
```

```
int remover(fila *f, tipo_elem *item){  
    if (!vazia(f)){  
        *item = (f->inicio)->info; //opcional  
        f->inicio = (f->inicio)->proximo;  
        if (f->inicio == NULL)  
            f->fim = NULL;  
        return 1;  
    }  
    return 0;  
}
```



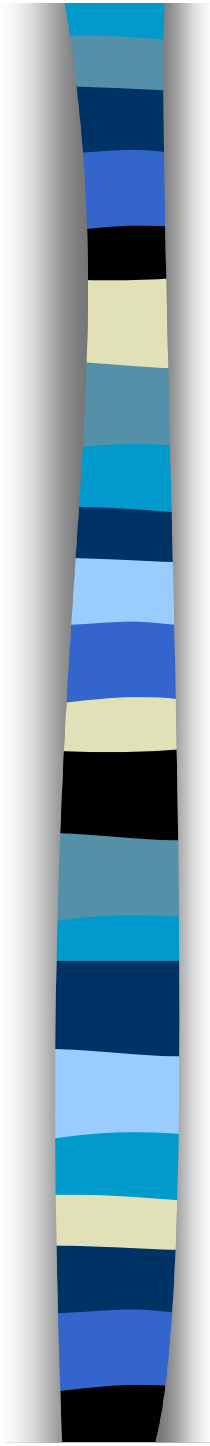


```
//Retorna o tamanho da fila.
```

```
int tamanho(fila *f){  
    no *tmp;  
    int qtdNos = 0;  
    if (!vazia(f)){  
        tmp = f->inicio;  
        while (tmp != NULL){  
            qtdNos++;  
            tmp = tmp->proximo;  
        }  
    }  
    return qtdNos;  
}
```

```
//Mostra o começo da fila sem remover o item. Retorna true (1)  
//se operação realizada com sucesso, false (0) caso contrário.
```

```
int comeco_fila (fila *f, tipo_elem *item){  
    if (!vazia(f)){  
        *item = (f->inicio)->info;  
        return 1;  
    }  
    return 0;  
}
```



```
//imprime a fila
void imprime(fila *f){
    no *tmp;
    if (!vazia(f)){
        printf("\nFila: ");
        tmp = f->inicio;
        while (tmp != NULL){
            printf("%d ", tmp->info);
            tmp = tmp->proximo;
        }
        return;
    }
    printf("\nFila vazia!\n");
}
```