Fila

ECM404

Fila

- Armazena elementos que estão em ***espera*** para serem processados.

  Em computação, uma fila serve para:

  - Armazenar eventos do sistema a serem processados;
  - Controlar o acesso simultâneo a recursos compartilhados (impressora, rede, disco, entre outros);
  - Controlar o processo de recebimento de mensagens em um programa (servidor WEB);
  - Aplicações específicas em alguns algoritmos (busca em largura, compressão Huffmann, entre outros).

Fila

- FIFO: First In First Out

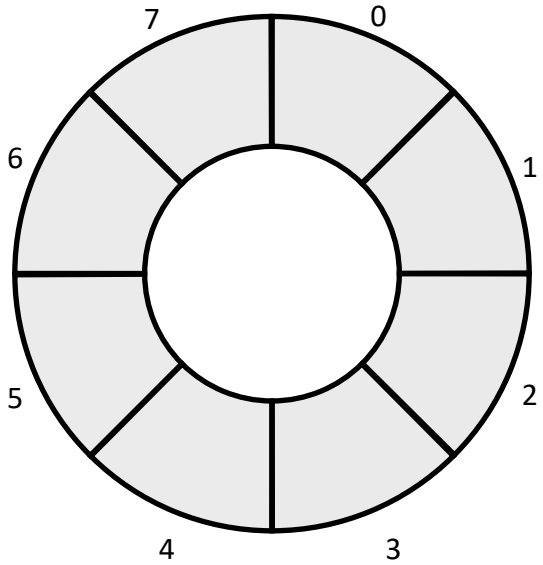
A operação de inserção coloca um elemento no ***final*** da fila enquanto que a operação de retirada retira o ***primeiro*** elemento da fila;

Estudaremos uma fila implementada com um vetor e acessada de forma circular

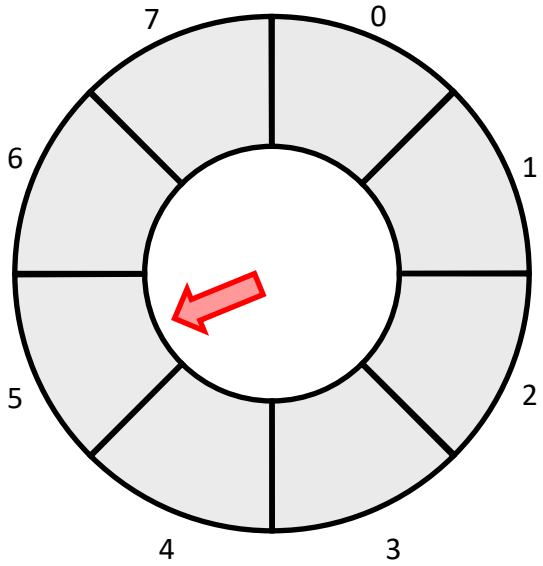
Fila Circular

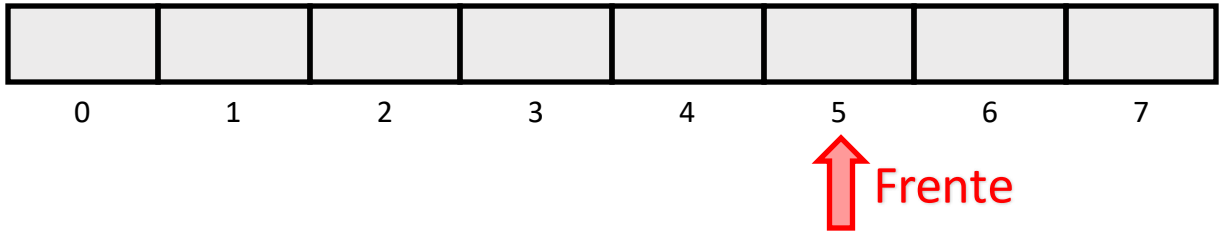
- Fila circular:
  - Itens são inseridos na posição “***trás***”;
  - Itens são retirados na posição ***“frente”***;
  - A fila está vazia quando a contagem do número de elementos é ***0***;
  - A fila está cheia quando a contagem do número de elementos é o ***tamanho total*** da fila;

Fila Circular – Criação da Fila

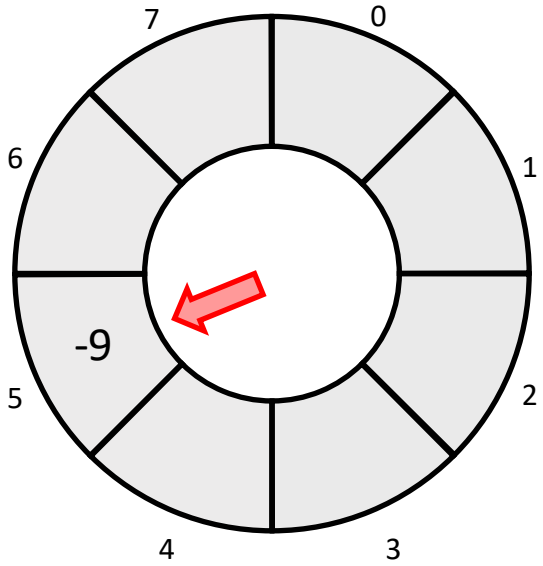


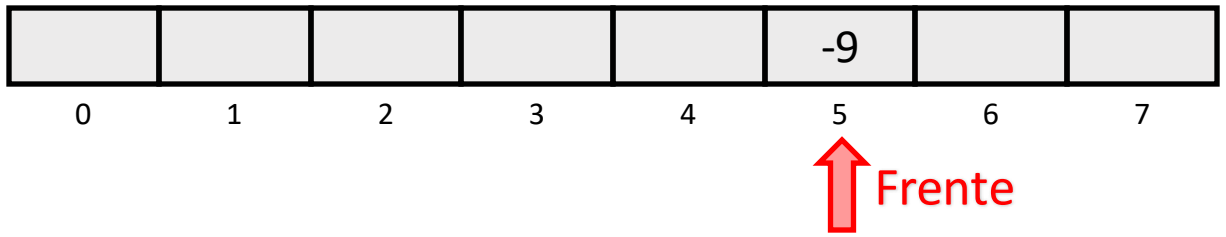
Fila Circular – Criação da Fila



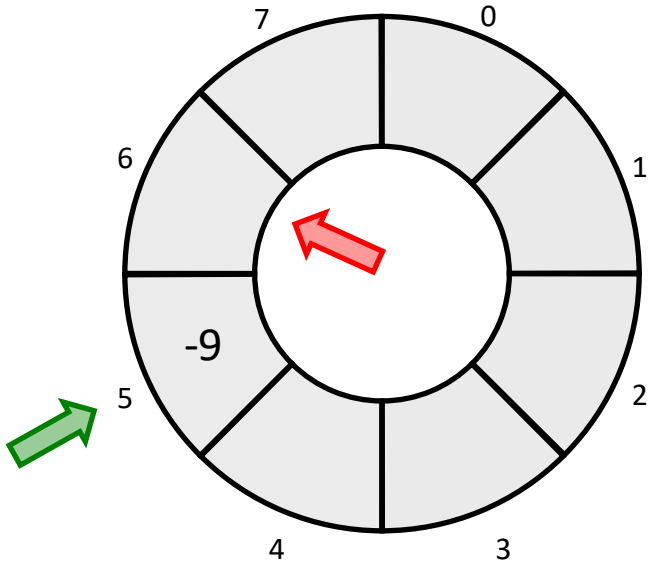


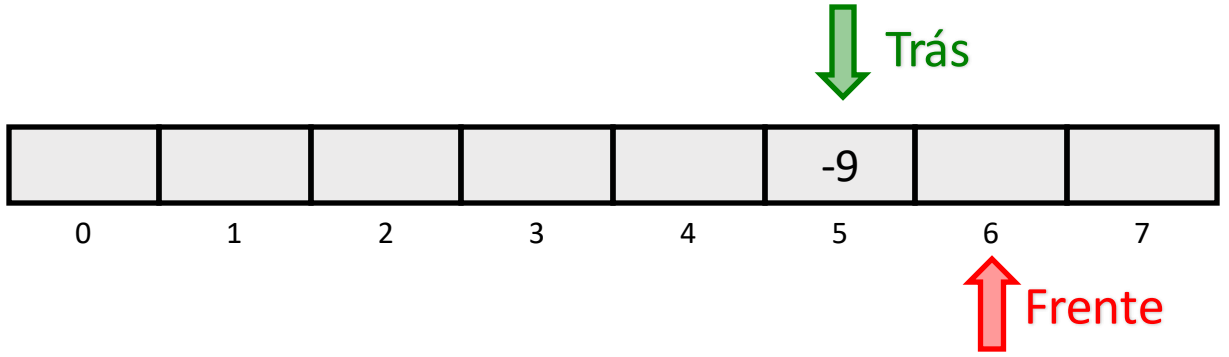
Fila Circular – Inserção



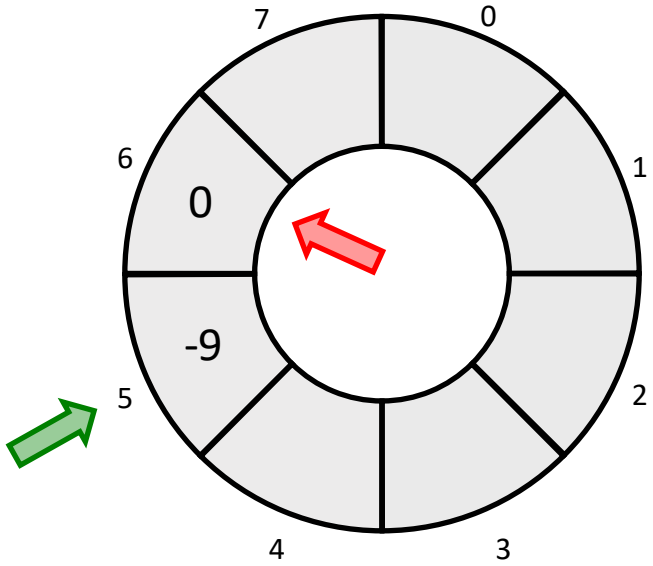


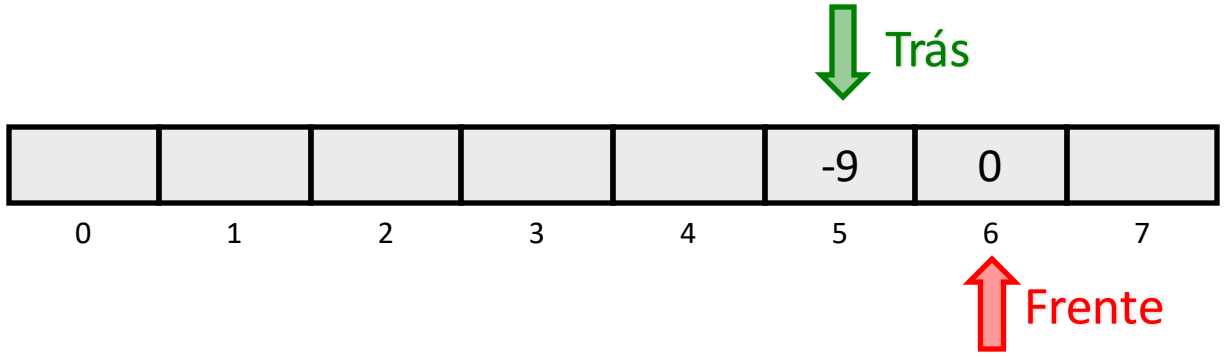
Fila Circular – Inserção



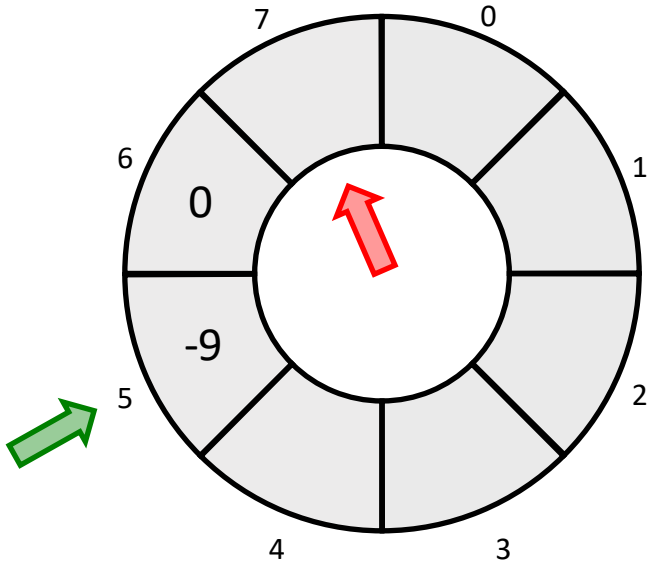


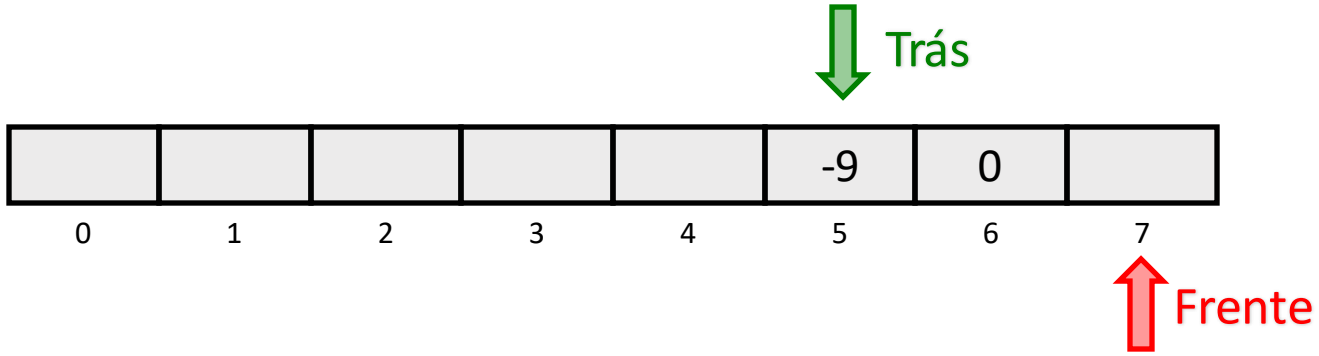
Fila Circular – Inserção



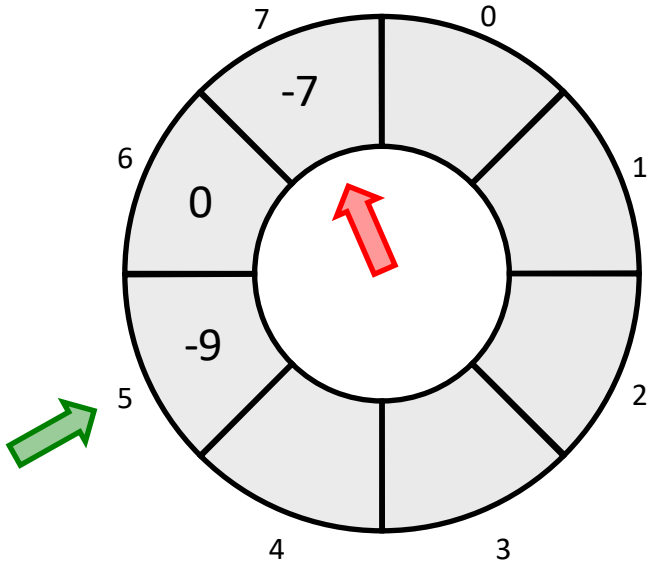


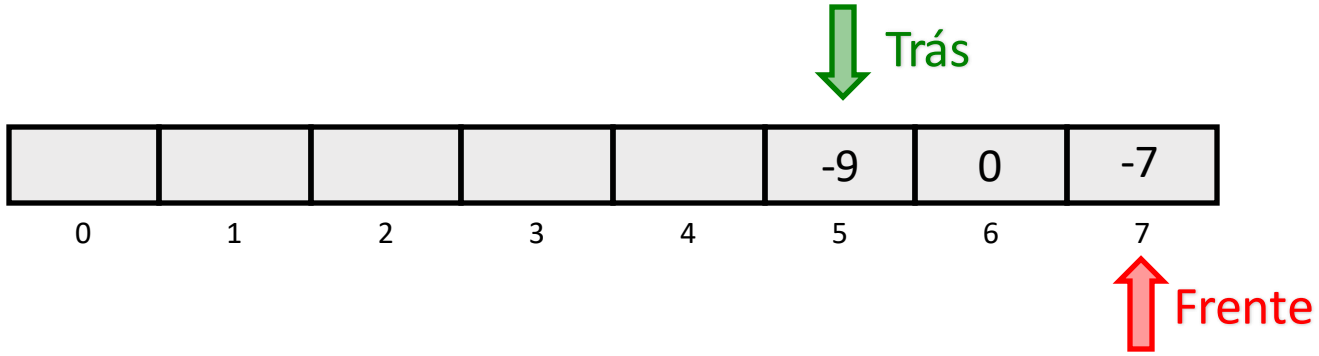
Fila Circular – Inserção



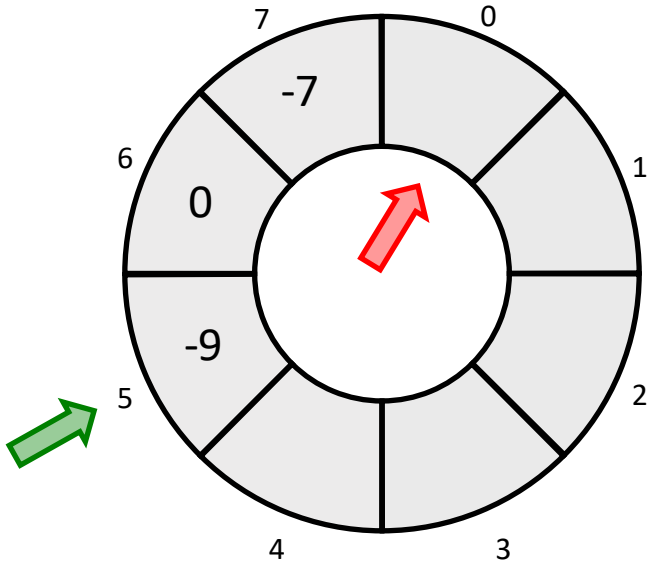


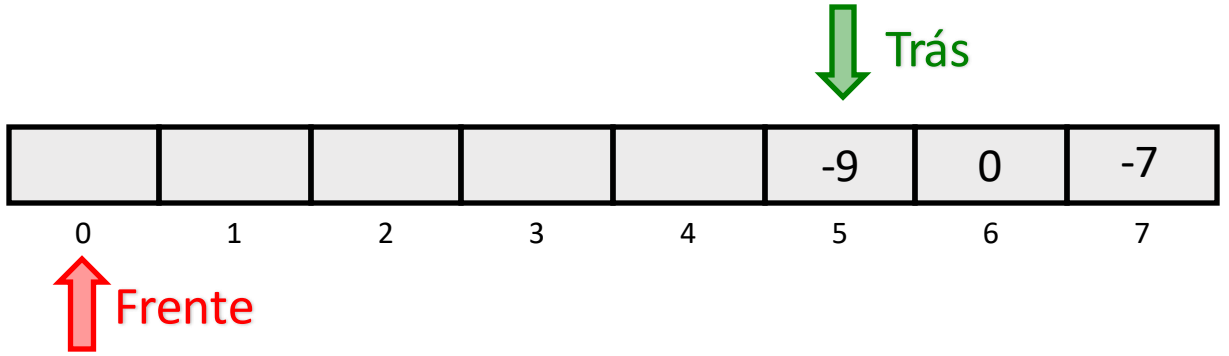
Fila Circular – Inserção



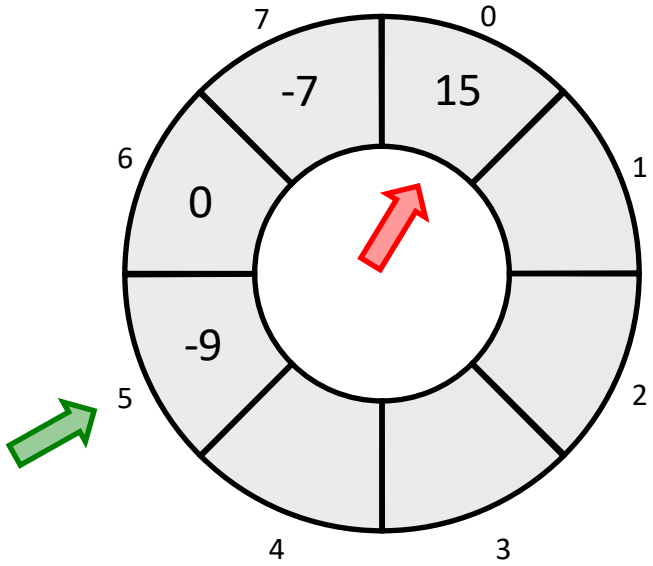


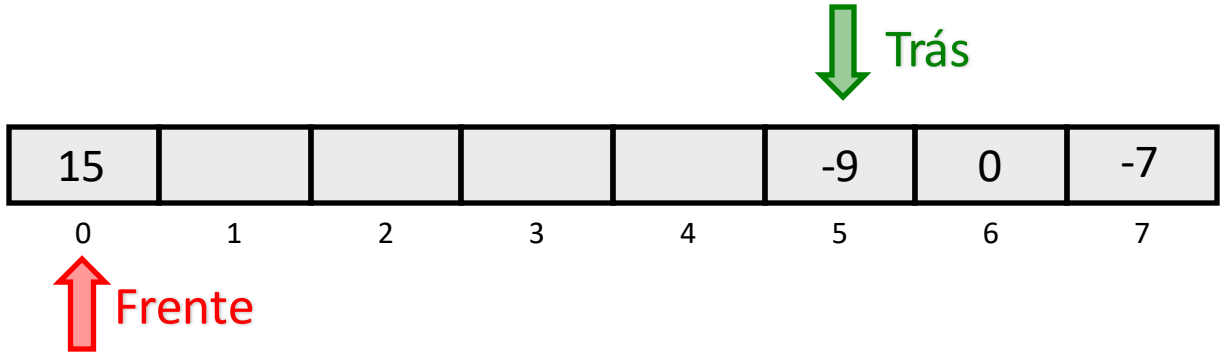
Fila Circular – Inserção



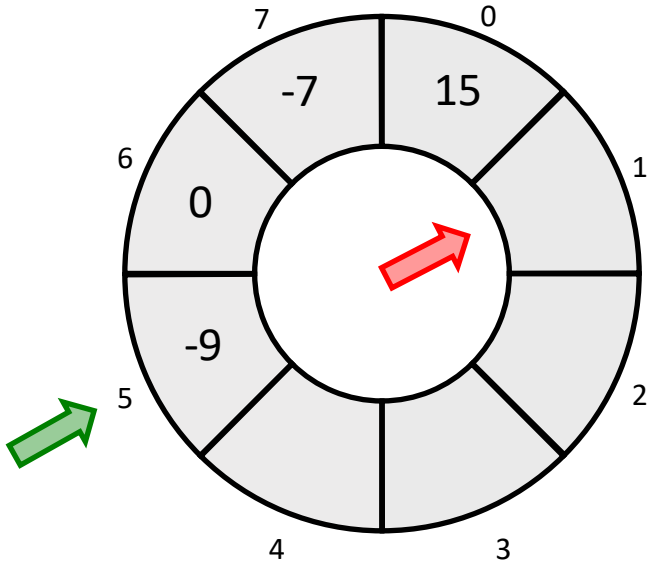


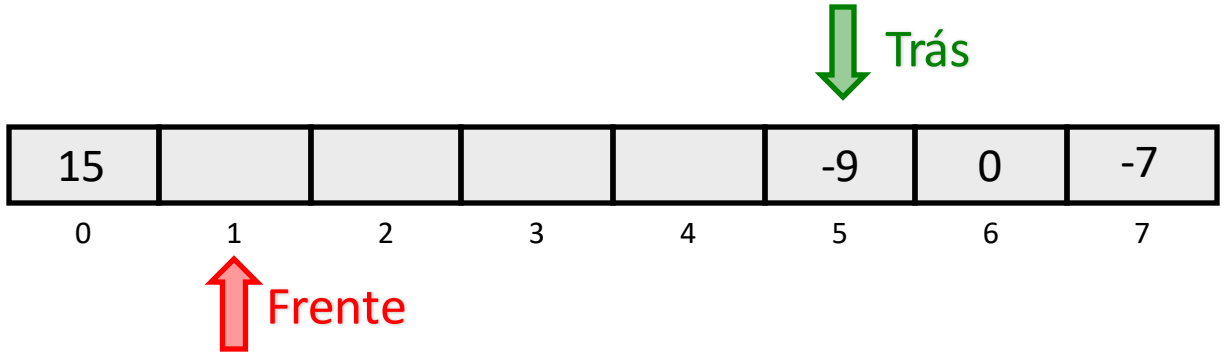
Fila Circular – Inserção



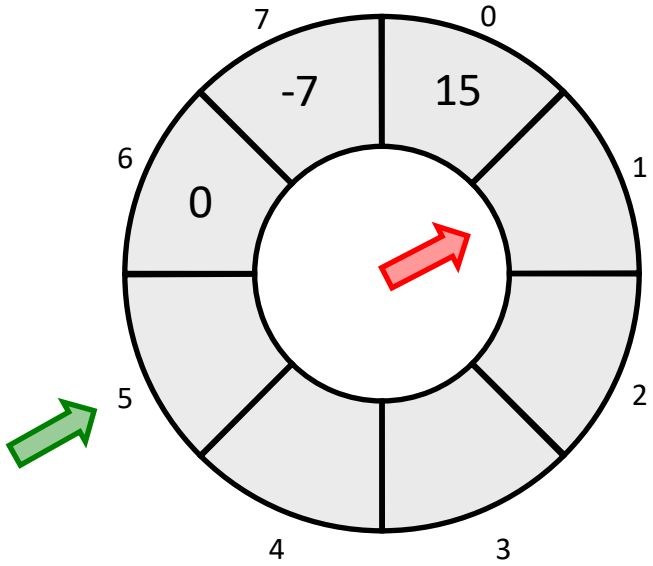


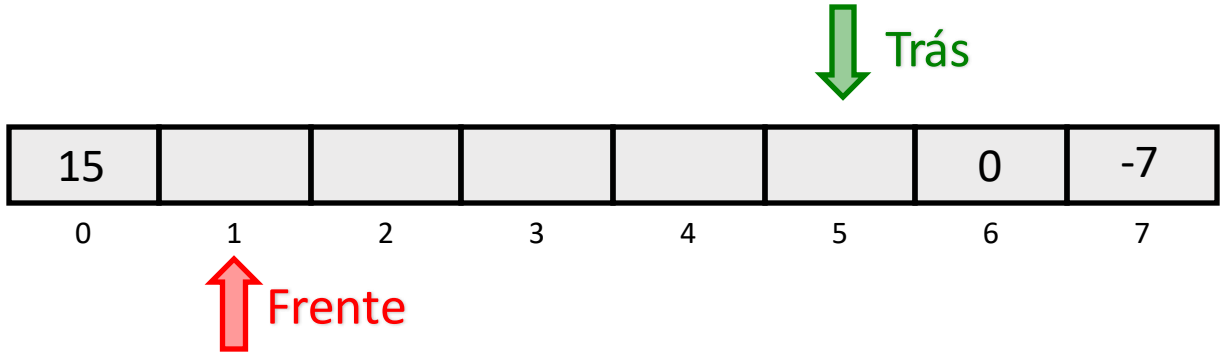
Fila Circular – Inserção



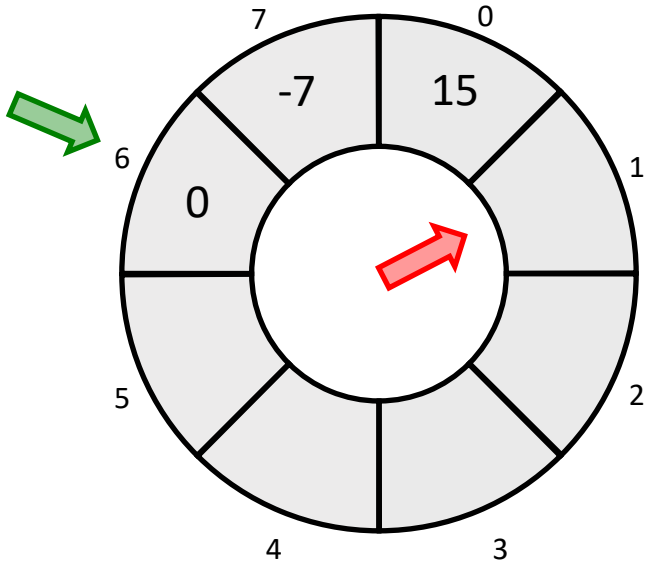


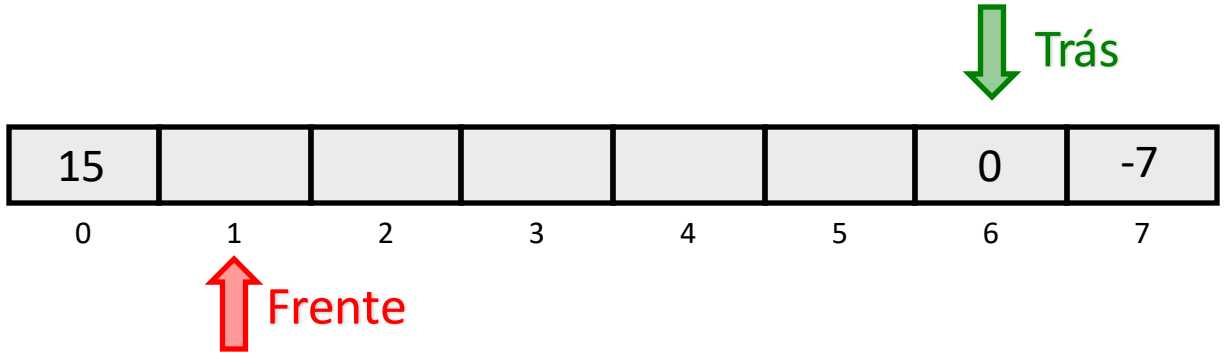
Fila Circular – Remoção





Fila Circular





Fila – Funções

```
//cria uma fila nova
void CriarFila(TFila *f);

//insere elemento
int InserirNaFila(TFila *f, Dados dados);

//retira elemento
int RetirarDaFila(TFila *f, Dados *dados);

//indica se a fila está vazia
int FilaVazia(TFila f);

//indica se a fila está cheia
int FilaCheia(TFila f);

//retorna o tamanho da fila
int QuantidadeNaFila(TFila f);
```

Lista Ligada – CriarFila

```
void CriarFila(TFila* f) {
 /* define valores iniciais das propriedades da fila */
    f->quantidade = 0;
    f->frente = 0;
    f->tras = -1;
}
```

Lista Ligada – InserirNaFila

```
int InserirNaFila(TFila *f, Dados dados) {
    int fc = FilaCheia(*f);
    if (!fc) {
       /* incrementa a posição "trás" no buffer circular*/
       f->tras = (f->tras + 1) % MAX_FILA;

       /* insere o elemento no fim da fila */
       f->elemento[f->tras] = dados;

       /* incrementa a quantidade */
       f->quantidade++;
    }
    return !fc; /* retorna se conseguiu inserir o dado */
}
```

Lista Ligada – RetirarDaFila

```
int RetirarDaFila(TFila *f, Dados *dados) {
    int fv = FilaVazia(*f);
    if (!fv) {
       /* recupera o dado do início da fila */
       *dados = f->elemento[f->frente];

       /* incrementa a posição "frente" no buffer circular*/
       f->frente = (f->frente +1) % MAX_FILA;

       /* decrementa a quantidade */
       f->quantidade--;
    }
    return !fv; /* retorna se conseguiu remover o dado */
}
```

Lista Ligada – Verificação

```
int FilaVazia(TFila f) {
    return f.quantidade == 0;
}


int FilaCheia(TFila f) {
    return f.quantidade == MAX_FILA;
}


int QuantidadeNaFila(TFila f) {
    return f.quantidade;
}
```

Bibliotecas

- São arquivos que contém estruturas de dados e sub-rotinas destinadas à resolução de um determinado problema;
- São criados dois arquivos:
  - Arquivo de cabeçalho (.h):
    - Estruturas de dados;
    - Protótipos das sub-rotinas que serão disponibilizadas pela biblioteca.
  - Arquivo de código (.c):
    - Código fonte das sub-rotinas;

Bibliotecas – Fila.h

```
#ifndef FILA_H_INCLUDED
#define FILA_H_INCLUDED

// Todo o conteúdo do arquivo .h vai aqui....

#endif // FILA_H_INCLUDED
```

Bibliotecas – Fila.h

```c
#ifndef FILA_H_INCLUDED
#define FILA_H_INCLUDED

#define MAX 20
#define MAX_FILA 10

// Definição de tipos

typedef struct {
    float valor;
    char texto[MAX];
} Dados;

typedef struct {
    Dados elemento[MAX_FILA];
    int quantidade, frente, tras;
} TFila;
```

Bibliotecas – Fila.h

```
//Protótipos de funções

//cria uma fila nova
void CriarFila(TFila *f);
//insere elemento
int InserirNaFila(TFila *f, Dados dados);
//retira elemento
int RetirarDaFila(TFila *f, Dados *dados);
//indica se a fila está vazia
int FilaVazia(TFila f);
//indica se a fila está cheia
int FilaCheia(TFila f);
//retorna o tamanho da fila
int QuantidadeNaFila(TFila f);

#endif // FILA_H_INCLUDED
```

Bibliotecas – Fila.c

- Para que sejam utilizadas as funções nativas da linguagem C é necessário incluir as bibliotecas no arquivo .c.
- Além disso, faz-se necessário incluir o arquivo .h da biblioteca que está sendo criada para que os protótipos e tipos de dados criado possam ser utilizados no arquivo.

```
#include <stdio.h>
#include <stdlib.h>
#include "fila.h"
```

Compilando as Bibliotecas

- Após a criação dos arquivos da biblioteca, é necessário que estes arquivos sejam compilados juntamente com o programa principal.

```
gcc programa_principal.c biblioteca.h biblioteca.c
```

EXERCÍCIO

- Abra a pasta Downloads no VS Code
  - Utilize o comando git clone https://github.com/ECM404/listaX.git --recursive
  - Entre na pasta utilizando o comando cd listaX