# ALGORITMOS E LÓGICA DE PROGRAMAÇÃO II

Prof. Wilson Lourenço

wilson.slourenco@sp.senac.br

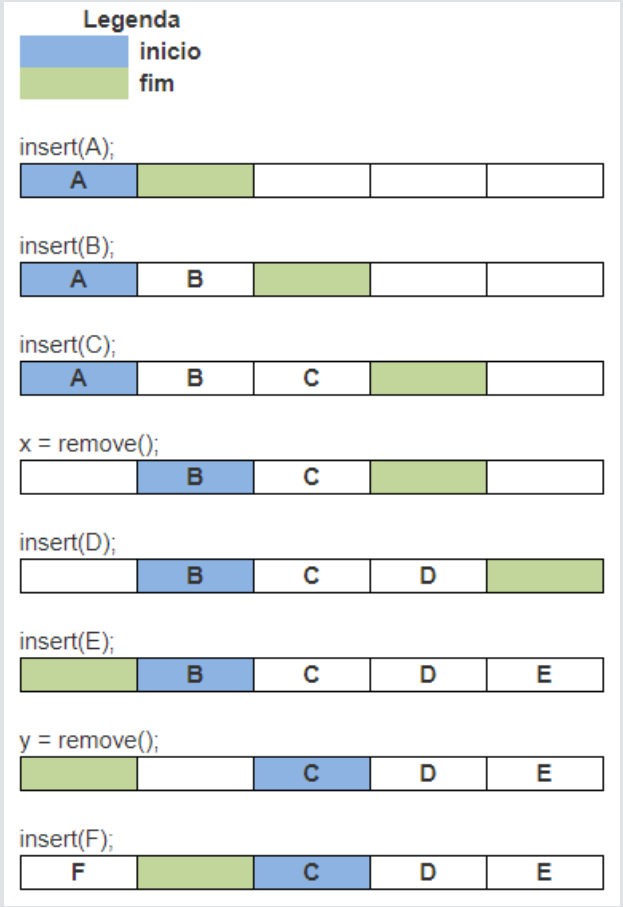
# FILA CIRCULAR

**Fila Circular**

Em sua forma mais simples a fila é uma estrutura de dados fácil de programar, entretanto, percebe-se rapidamente que a aparente simplicidade esconde algumas complicações.

Como a questão da representação na memória, pode-se representar a fila como um vetor ou uma lista encadeada, no primeiro caso um problema singular ocorre.

Quando se usa vetores para implementar filas, podemos contornar este problema transformando essas filas em um vetor circular, de forma que ao acessar o ultimo elemento do vetor, continua-se a partir do primeiro, obtendo assim a fila circular, resultando na situação a seguir:


Legenda
inicio
fim
insert(A);
A
insert(B);
A
B
insert(C);
A
B
C
x = remove();
B
C
insert(D);
B
C
D
insert(E);
B
C
D
E
y = remove();
C
D
E
insert(F);
F
C
D
E

O código modifica o exemplo anterior para incluir o suporte ao buffer circular, a necessidade de indicar o status de fila cheia, as variáveis que apontam para o inicio e fim, podem apontar para a mesma posição do vetor em duas situações, quando a fila está vazia ou cheia:

Status = livre

| | | | | |
|---|---|---|---|---|
| inicio | | | | |
| | | | | |
| fim | | | | |

Status = cheia

| | | | | |
|---|---|---|---|---|
| | | | inicio | |
| F | G | H | D | E |
| | | | fim | |

Vídeo:

https://www.youtube.com/watch?v=nYiG_JKY6Ik

Exemplo 01: (Ex_FCircular01.java)

```
public class Ex_FCircular01 {

    static final int MAX= 4; // numero maximo de elementos na fila

    // cria uma fila vazia
    static int comeco = 0;   // comeco da fila
    static int tamanho = 0;   // tamanho da fila (numero de elementos)
    static int queue[] = new int[MAX];   // vetor da fila
```

```
public static void main(String[] args) {

        int i; // contador

        inserir(1);
        inserir(10);
        inserir(100);
        inserir(1000);

        System.out.println("");

        remover();
        inserir(6);
```

```
        remover();
        inserir(60);

        remover();
        inserir(80);

        System.out.println("");

        //// mostra fila na tela ////
        for(i = 0; i < MAX; i++)
            System.out.println("fila[" + i + "] = " + queue[i]);
    }
```

```
static void inserir( int elemento )
{
   //// checa se a fila esta cheia ////
   if(tamanho == MAX)
        System.out.println("\nfila cheia\n");
   else {
        //Para tornar a fila circular
        queue[((comeco + tamanho) % MAX) ] = elemento;

        System.out.println("Valor "  + elemento
                + " inserido no índice "
                + ((comeco + tamanho) % MAX) + " da fila");
```

```
        //incrementa tamanho da fila (elemento foi inserido)
        tamanho ++;
    }
} // fim funcao

static void remover()
{
    //// checa se a fila esta vazia ////
    if( tamanho == 0 )
        System.out.println("\nfila vazia\n");
```

```
else {
        //Apaga o primeiro elemento da fila deslocando o
        //ponteiro do comeco para proximo elemento ////
        comeco ++;
        //// decrementa o contador de tamanho (um valor foi
removido)
        tamanho --;
      }
   } // fim funcao
}
```

Exemplo 02: (Ex_FCircular02.java)

```
public class Fila {

    int dados[] = new int[5];
    int inicio;
    int fim;
}
```

```
public class Ex_FCircular02 {

    static final int TAMANHO = 5;

    public static void main(String[] args) {

        Fila minhaFila  = new Fila();

       criaFila(minhaFila);

       enQueue(minhaFila,1);

        System.out.println("Valor 1 adicionado a fila!\n");
```

```
enQueue(minhaFila,2);
 System.out.println("Valor 2 adicionado a fila!\n");

enQueue(minhaFila,3);
 System.out.println("Valor 3 adicionado a fila!\n");

 enQueue(minhaFila,4);
 System.out.println("Valor 4 adicionado a fila!\n");

enQueue(minhaFila,5);
System.out.println("Valor 5 adicionado a fila!\n\n");

 System.out.println("Tentativa de adicionar o valor 6 a fila!\n");
```

```
        enQueue(minhaFila,6); // erro fila cheia

         System.out.println("Valor " + deQueue(minhaFila) + "
removido da fila!\n");

         System.out.println("Valor " + deQueue(minhaFila) + "
removido da fila!\n\n");

        enQueue(minhaFila,7);
         System.out.println("Valor 7 adicionado a fila!\n");

        enQueue(minhaFila,8);
         System.out.println("Valor 8 adicionado a fila!\n\n");
```

```
System.out.println("Valor " + deQueue(minhaFila) + " removido da fila!\n");

    System.out.println("Valor " + deQueue(minhaFila) + " removido da fila!\n");

    System.out.println("Valor " + deQueue(minhaFila) + " removido da fila!\n");

    System.out.println("Valor " + deQueue(minhaFila) + " removido da fila!\n");

     System.out.println("Valor " + deQueue(minhaFila) + " removido da fila!\n\n");
```

```
System.out.println("Tentativa de remover valor da fila!");

        System.out.println(deQueue(minhaFila)); // erro fila vazia

    }

    static void enQueue(Fila f, int dado)
    {
        if((f.inicio == f.fim + 1) || (f.inicio == 0 && f.fim ==
TAMANHO-1))
            System.out.println("\nErro: fila cheia\n\n\n");
```

```
else
        {
            if(f.inicio == -1)
            f.inicio = 0;
            f.fim=(f.fim+1) % TAMANHO;
            f.dados[f.fim] = dado;
        }
    }

    static void criaFila(Fila f)
    {
            f.inicio=-1;
            f.fim=-1;
    }
```

```
static int deQueue(Fila f)
    {
        int dado;
        if(f.inicio == -1)
        {
            System.out.println("\nErro: fila vazia\n\n");
            return(0);
        }
        else
        {
            dado=f.dados[f.inicio];
```

```
if (f.inicio == f.fim)
            {
                f.inicio = f.fim = -1;
            }
            else
            {
                f.inicio = (f.inicio + 1) % TAMANHO;
            }
        }
        return dado;
    }
```

```
    static void imprimeDados(Fila f)
    {
        System.out.println("\n[");
        int cont;
        for(cont=0; cont<TAMANHO; cont++)
            System.out.println(f.dados[cont]);
        System.out.println("]");
    }
}
```

Exemplo 03: (Ex_FCircular3.java)

```
package Ex_FCircular03;

import java.util.Arrays;

public class Ex_FCircular03 {

    public static void main(String[] args) {

        /*CircularQueue<Integer> circularQueue = new CircularQueue(8);
```

```
circularQueue.enqueue(15);
circularQueue.enqueue(16);
circularQueue.enqueue(17);
circularQueue.enqueue(18);
circularQueue.enqueue(19);
circularQueue.enqueue(20);
circularQueue.enqueue(21);
circularQueue.enqueue(22);
*/

CircularQueue<String> circularQueue = new CircularQueue(8);
```

```
circularQueue.enqueue("Banana");
        circularQueue.enqueue("Maça");
        circularQueue.enqueue("Pera");
        circularQueue.enqueue("Uva");
        circularQueue.enqueue("Abacaxi");
        circularQueue.enqueue("Melão");
        circularQueue.enqueue("Melancia");
        circularQueue.enqueue("Goiaba");


        System.out.println("Fila circular cheia: " + circularQueue);
```

```
 System.out.print("\nDesenfileirando elemento:");
        System.out.println(circularQueue.dequeue() + " ");


        circularQueue.enqueue("Mexerica");
        System.out.println("\nApós enfileirar o elemento
Mexerica\n");

 //circularQueue.enqueue(23);
        System.out.println("\nApós enfileirar o elemento 23\n");

        System.out.println(circularQueue + "\n");
    }
```

```
class CircularQueue<E> {
    private int currentSize;
    private E[] circularQueueElements;
    private int maxSize;
    private int rear;
    private int front;

 public CircularQueue(int maxSize) {
        this.maxSize = maxSize;
        circularQueueElements = (E[]) new Object[this.maxSize];
        currentSize = 0;
        front = -1;
        rear = -1;
    }
```

```
public void enqueue(E item) throws QueueFullException {
        if (isFull()) {
            throw new QueueFullException("A fila está cheia!. O
elemento não pode ser adicionado!");
        }
        else{
            rear = (rear + 1) % circularQueueElements.length;
            circularQueueElements[rear] = item;
            currentSize++;

            if (front == -1) {
                        front = rear;
            }
        }
    }
```

```
public E dequeue() throws QueueEmptyException {
        E deQueuedElement;
        if (isEmpty()) {
            throw new QueueEmptyException("Fila circular vazia!.
O Elemento não pode ser removido!");
        }
        else {
            deQueuedElement = circularQueueElements[front];
            circularQueueElements[front] = null;
            front = (front + 1) % circularQueueElements.length;
            currentSize--;
        }
        return deQueuedElement;
    }
```

```
 public boolean isFull() {
        return (currentSize == circularQueueElements.length);
    }

    public boolean isEmpty() {
        return (currentSize == 0);
    }

    @Override
    public String toString() {
        return "CircularQueue [" +
Arrays.toString(circularQueueElements) + "]";
    }
}
```

```
class QueueFullException extends RuntimeException {

    public QueueFullException() {
        super();
    }

    public QueueFullException(String message) {
        super(message);
    }

}
```

```
class QueueEmptyException extends RuntimeException {

    public QueueEmptyException() {
        super();
    }

    public QueueEmptyException(String message) {
        super(message);
    }
}
```

O JAVA usa uma convenção de nomenclatura para as letras de identificação de generics, sendo:

- E - Elemento
- K - Chave
- N - Número
- T - Tipo
- V - Valor
- S, U - Tipos adicionais

Quizziz:

Joinmyquizziz.com

# Dicas para Estudo

**Seja “CURIOSO”:**

- Procure revisar o que foi estudado.
- Pesquise as referências bibliográficas.

**Seja “ANTENADO”:**

- Leia a próxima aula.

**Seja “COLABORATIVO”:**

- Traga assuntos relevantes para a sala de aula.
- Participe da aula.
- Proponha discussões relevantes sobre o conteúdo.

Prof. Me. Wilson Lourenço


Dúvidas?
Não mais..