



# 4

# Estruturas de dados do tipo pilha e fila

As estruturas de dados do tipo pilha e fila são consideradas listas especializadas por possuírem características próprias. Mas, apesar dessas características, ambas possuem operações como inserir um elemento, excluir um elemento, encontrar o maior, encontrar o menor, contar os elementos, alterar um elemento, buscá-lo, buscar o sucessor e buscar o predecessor.

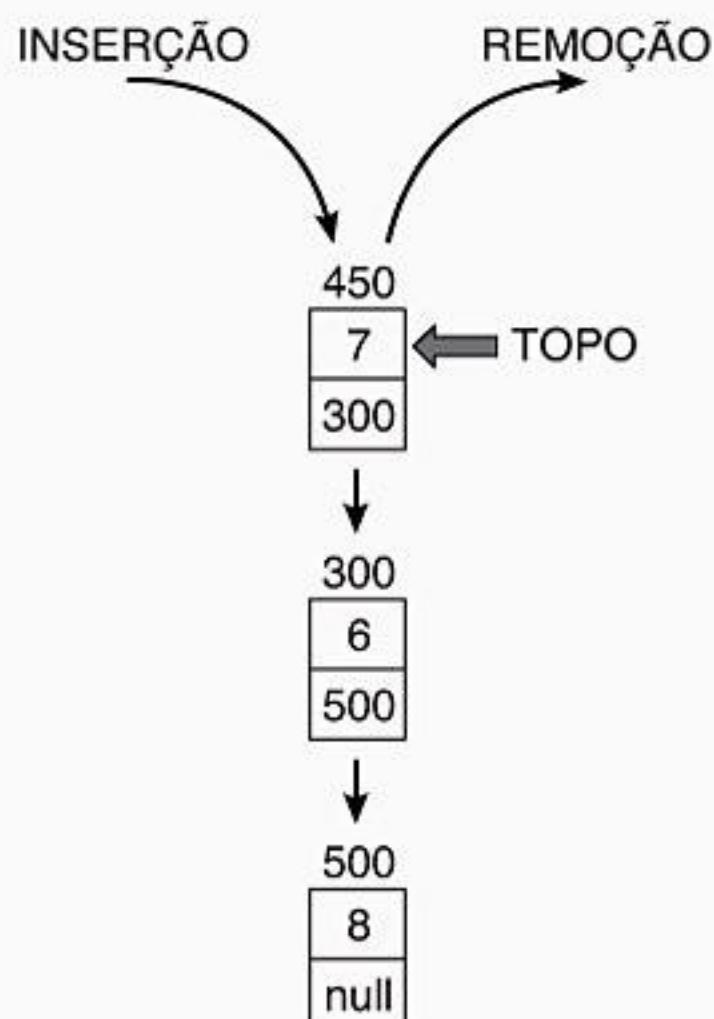
Essas duas estruturas de dados representam conjuntos de dados que estão organizados em ordem linear. Quando essas estruturas são representadas por arranjos, ou seja, quando é feita a utilização de vetores nas representações, tem-se o uso de endereços contíguos de memória do computador e a ordem linear é determinada pelos índices dos vetores, o que em algumas situações exige maior esforço computacional. Tais representações denominam-se pilha e fila estáticas. Quando as estruturas pilha e fila são representadas por elementos que, além de conter o dado, possuem também um ponteiro para o próximo elemento, ou seja, elementos encadeados, têm-se representações denominadas pilha e fila dinâmicas.

Quando um elemento de uma pilha ou de uma fila contém apenas um dado primitivo, como um número, chama-se pilha ou fila homogênea, e quando um elemento de uma pilha ou fila contém um dado composto, como o nome e o salário de um funcionário, chama-se pilha ou fila heterogênea.

As próximas seções mostram as estruturas pilha e fila com ilustração, implementação dinâmica e análise.

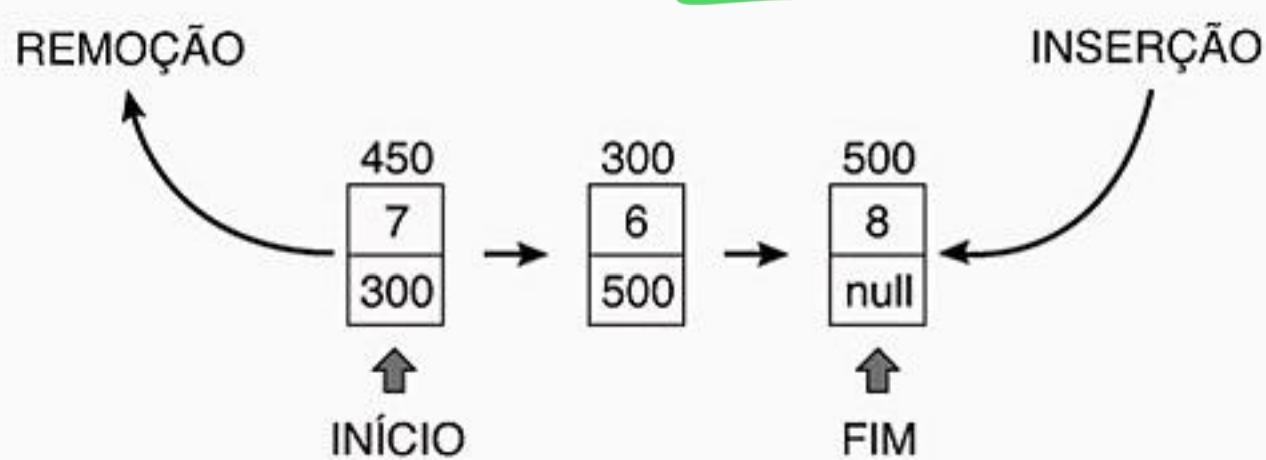
A Figura 4.1 ilustra uma estrutura de dados do tipo pilha, e a Figura 4.2 ilustra uma estrutura de dados do tipo fila.

Figura 4.1 Estrutura de dados do tipo pilha



Jámos, a inserção e remoção acontecendo no topo.

Figura 4.2 Estrutura de dados do tipo fila



→ inserir no fim  
e remove no início

Pilha

= o primeiro elemento inserido  
será o último a ser removido.

A estrutura denominada pilha é considerada do tipo FILO (*first in last out*), ou seja, o primeiro elemento inserido será o último a ser removido. Nessa estrutura, cada elemento armazena um ou vários dados (estrutura homogênea ou heterogênea, respectivamente) e um ponteiro para o próximo elemento, permitindo o encadeamento e mantendo a estrutura linear. Nesse tipo de estrutura serão abordadas as seguintes operações: inserir na pilha, consultar toda a pilha, remover e esvaziá-la. Qualquer estrutura desse tipo possui um ponteiro denominado TOPO, no qual todas as operações de inserção e remoção acontecem. Assim, as operações ocorrem sempre na mesma extremidade da estrutura.

Veja a ilustração a seguir, com endereços de memória meramente ilustrativos.



ILUSTRAÇÃO

**1<sup>a</sup> operação**  
**A pilha está vazia**

topo  
null

**2<sup>a</sup> operação**  
**Inserção do nº 9 na pilha**

topo      800  
800      9  
            null

**3<sup>a</sup> operação**  
**Inserção do nº 3 na pilha**

topo      650  
650      3  
            800  
                ↓  
800      9  
            null

**4<sup>a</sup> operação**  
**Inserção do nº 5 na pilha**

topo      750  
750      5  
            650  
                ↓  
650      3  
            800  
                ↓  
800      9  
            null

**5<sup>a</sup> operação**  
**Remoção da pilha**

topo      650  
650      3  
            800  
                ↓  
800      9  
            null

**6<sup>a</sup> operação**  
**Remoção da pilha**

topo      800  
800      9  
            null



```
import java.util.*;
public class Pilha
{
    //Definindo a classe que representará
    //cada elemento da pilha
    private static class PILHA
    {
        public int num;
        public PILHA prox;
    }

    public static void main(String args[])
    {
        Scanner entrada = new Scanner(System.in);
        // a pilha está vazia, logo,
        // o objeto topo tem o valor null
        // as operações de inserção e remoção
        // acontecem no TOPO
        PILHA topo = null;
        // o objeto aux é um objeto auxiliar
        PILHA aux;
        // apresentando o menu de opções
        int op;
        do
        {
            System.out.println("\nMENU DE OPÇÕES\n");
            System.out.println("1 - Inserir na pilha");
            System.out.println("2 - Consultar toda a
                → pilha");
            System.out.println("3 - Remover da pilha");
            System.out.println("4 - Esvaziar a pilha");
            System.out.println("5 - Sair");
            System.out.print("Digite sua opção: ");
            op = entrada.nextInt();
            if (op < 1 || op > 5)
                System.out.println("Opção inválida!!!");
            if (op == 1)
            {
                System.out.println("Digite o número a ser
                    → inserido na pilha: ");
                PILHA novo = new PILHA();
                novo.num = entrada.nextInt();
                novo.prox = topo;
                topo = novo;
            }
        } while (topo != null);
    }
}
```

```
System.out.println("Número inserido na  
→ pilha!!");  
}  
if (op == 2)  
{  
    if (topo == null)  
    {  
        // a pilha está vazia  
        System.out.println("Pilha vazia!!");  
    }  
    else  
    {  
        // a pilha contém elementos e  
        // estes serão mostrados  
        // do último inserido ao primeiro  
        System.out.println("\nConsultando a  
→ pilha\n");  
        aux = topo;  
        while (aux != null)  
        {  
            System.out.print(aux.num+" ");  
            aux = aux.prox;  
        }  
    }  
}  
if (op == 3)  
{  
    if (topo == null)  
    {  
        // a pilha está vazia  
        System.out.println("Pilha vazia!!");  
    }  
    else  
    {  
        // a pilha contém elementos  
        // e o último elemento inserido  
        // será removido  
        System.out.println("Número "+topo.num+"  
→ removido");  
        topo = topo.prox;  
    }  
}  
if (op == 4)  
{
```

```

        if (topo == null)
        {
            // a pilha está vazia
            System.out.println("Pilha vazia!!");
        }
        else
        {
            // a pilha será esvaziada
            topo = null;
            System.out.println("Pilha esvaziada");
        }
    }
    while (op != 5);
}

```

---



```

#include <iostream.h>
#include <conio.h>

//Definindo o registro que representará cada elemento
//cada elemento da pilha

struct PILHA
{
    int num;
    PILHA *prox;
};

void main()
{
    // a pilha está vazia, logo,
    // o ponteiro topo tem o valor null
    // as operações de inserção e remoção
    // acontecem no TOPO
    PILHA *topo = NULL;
    // o ponteiro aux é um ponteiro auxiliar
    PILHA *aux;
    // apresentando o menu de opções
    int op;
    do

```

```
{  
clrscr();  
cout<<"\nMENU DE OPÇÕES\n";  
cout<<"\n1 - Inserir na pilha";  
cout<<"\n2 - Consultar toda a pilha";  
cout<<"\n3 - Remover da pilha";  
cout<<"\n4 - Esvaziar a pilha";  
cout<<"\n5 - Sair";  
cout<<"\nDigite sua opção: ";  
cin>>op;  
if (op < 1 || op > 5)  
    cout<<"Opção inválida!!";  
if (op == 1)  
{  
    cout<<"Digite o número a ser inserido na  
    ↪ pilha: ";  
    PILHA *novo = new PILHA();  
    cin>>novo->num;  
    novo->prox = topo;  
    topo = novo;  
    cout<<"Número inserido na pilha!!";  
}  
if (op == 2)  
{  
    if (topo == NULL)  
    {  
        // a pilha está vazia  
        cout<<"Pilha vazia!!";  
    }  
    else  
    {  
        // a pilha contém elementos e  
        // estes serão mostrados  
        // do último inserido ao primeiro  
        cout<<"\nConsultando toda a pilha\n";  
        aux = topo;  
        while (aux != NULL)  
        {  
            cout<<aux->num<<" ";  
            aux = aux->prox;  
        }  
    }  
}  
if (op == 3)  
{
```

```
if (topo == NULL)
{
    // a pilha está vazia
    cout<<"Pilha vazia!!";
}
else
{
    // a pilha contém elementos
    // e o último elemento inserido
    // será removido
    aux = topo;
    cout<<"Número "<<topo->num<< " → removido";
    topo = topo->prox;
    delete(aux);
}
if (op == 4)
{
    if (topo == NULL)
    {
        // a pilha está vazia
        cout<<"Pilha vazia!!";
    }
    else
    {
        // a pilha será esvaziada
        aux=topo;
        while(aux!=NULL)
        {
            topo = topo->prox;
            delete(aux);
            aux=topo;
        }
        cout<<"Pilha esvaziada";
    }
}
getch();
}
while (op != 5);
}
```

## Análise da complexidade

A operação de inserção e remoção na pilha sempre realiza operações básicas, como a de atribuição, para atualizar o topo da pilha. Logo, são operações de tempo constante e gastam tempo  $O(1)$ .

Já a operação de consultar toda a pilha percorre todos os elementos armazenados nela. Considerando que uma pilha contém  $n$  elementos, o tempo de execução será  $O(n)$ .

A operação de esvaziamento da pilha consiste em remover todos os elementos dela. O tempo gasto nessa operação depende da linguagem de programação que está sendo utilizada. No caso da JAVA, não é necessário remover cada um dos elementos, apenas atualiza-se o ponteiro topo da pilha para nulo e as memórias alocadas serão desalocadas por um procedimento JAVA. Já na linguagem C/C++, é necessário desalocar cada um dos elementos da pilha, gastando tempo proporcional ao tamanho dela, ou seja,  $O(n)$ .

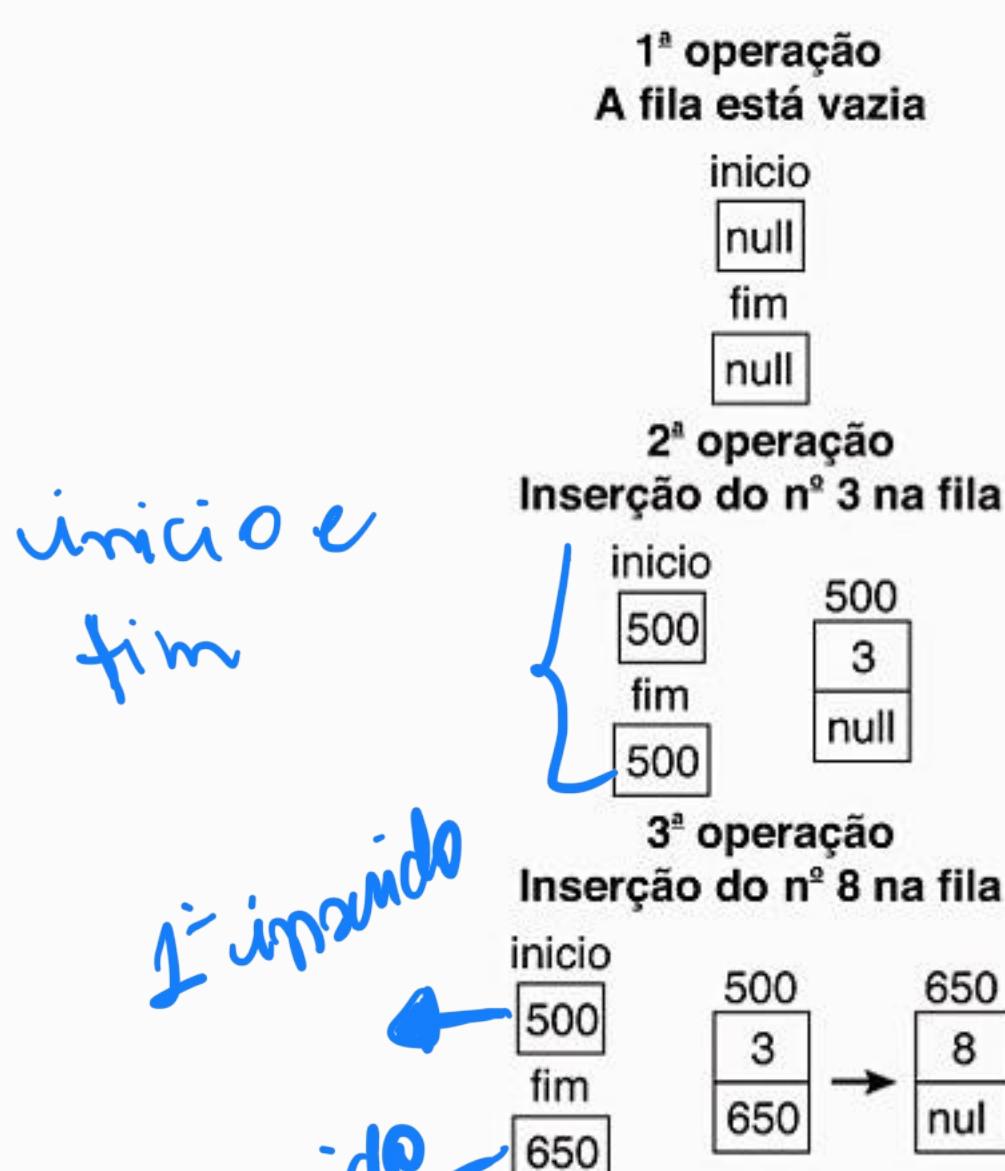
Fila  = primeiro elemento inserido será o primeiro a ser removido.

A estrutura denominada fila é considerada do tipo FIFO (First In First Out), ou seja, o primeiro elemento inserido será o primeiro a ser removido. Nessa estrutura, cada elemento armazena um ou vários dados (estrutura homogênea ou heterogênea, respectivamente) e um ponteiro para o próximo elemento, permitindo o encadeamento e mantendo a estrutura linear. Nesse tipo de estrutura serão abordadas as seguintes operações: inserir na fila, consultar toda a fila, remover e esvaziá-la. A estrutura do tipo fila possui um ponteiro denominado INICIO, onde as remoções acontecem, e um denominado FIM, onde as inserções acontecem. Assim, as operações ocorrem nas duas extremidades da estrutura.

Veja a ilustração a seguir, com endereços de memória meramente ilustrativos.



## ILUSTRAÇÃO



início e  
fim

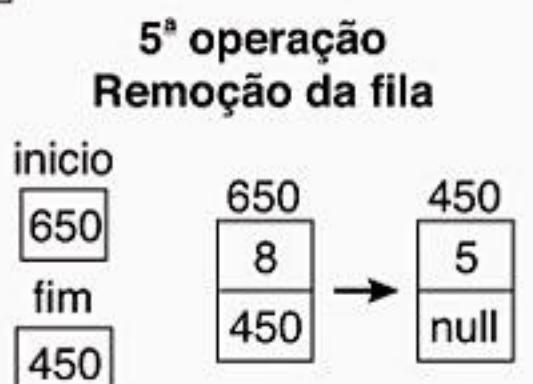
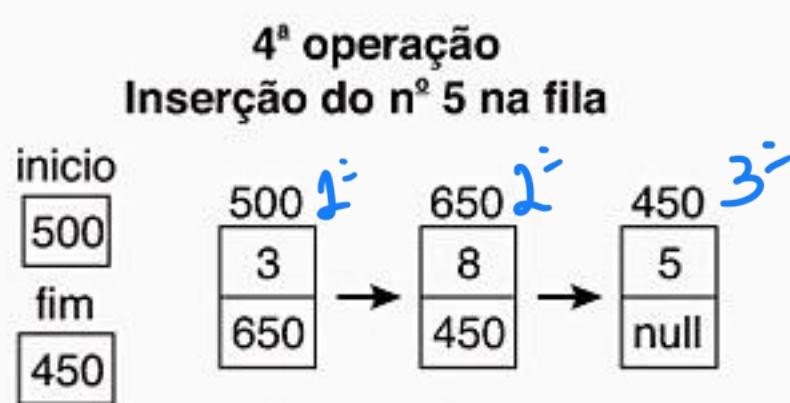
1- immuno

2' inside

A inserções  
acontecem  
no FIM

A remoção  
vacanteem

no INICIO



remova o 650,  
1º elemento  
remova o 450  
e depois o  
último.



```

import java.util.*;
public class Fila
{
    //Definindo a classe que representará
    //cada elemento da fila
    private static class FILA
    {
        public int num;
        public FILA prox;
    }

    public static void main(String args[])
    {
        Scanner entrada = new Scanner(System.in);
        // a fila está vazia, logo,
        // o objeto inicio tem o valor null
        // a operação de remoção acontece no INÍCIO
        // e a operação de inserção acontece no FIM
        FILA inicio = null;
        FILA fim = null;
        // o objeto aux é um objeto auxiliar
        FILA aux;
        // apresentando o menu de opções
    }
}

```

```
int op;
do
{
    System.out.println("\nMENU DE OPÇÕES\n");
    System.out.println("1 - Inserir na fila");
    System.out.println("2 - Consultar toda a fila");
    System.out.println("3 - Remover da fila");
    System.out.println("4 - Esvaziar a fila");
    System.out.println("5 - Sair");
    System.out.print("Digite sua opção: ");
    op = entrada.nextInt();
    if (op < 1 || op > 5)
        System.out.println("Opção inválida!!!");
    if (op == 1)
    {
        System.out.println("Digite o número a
➥ ser inserido na fila: ");
        FILA novo = new FILA();
        novo.num = entrada.nextInt();
        novo.prox = null;
        if (inicio == null)
        {
            // a fila está vazia
            // e o número inserido
            // será o primeiro e o último
            inicio = novo;
            fim = novo;
        }
        else
        {
            fim.prox = novo;
            fim = novo;
        }
        System.out.println("Número inserido na
➥ fila!!!");
    }
    if (op == 2)
    {
        if (inicio == null)
        {
            // a fila está vazia
            System.out.println("Fila vazia!!!");
        }
        else
        {
```

```
// a fila contém elementos e
// estes serão mostrados
// do primeiro inserido ao último
System.out.println("\nConsultando toda a
➥ fila\n");
aux = inicio;
while (aux != null)
{
    System.out.print(aux.num+" ");
    aux = aux.prox;
}
}
if (op == 3)
{
    if (inicio == null)
    {
        // a fila está vazia
        System.out.println("Fila vazia!!!");
    }
    else
    {
        // a fila contém elementos
        // e o primeiro elemento
        // inserido será removido
        System.out.println("Número "+inicio.num+
➥ removido");
        inicio = inicio.prox;
    }
}
if (op == 4)
{
    if (inicio == null)
    {
        // a fila está vazia
        System.out.println("Fila vazia!!!");
    }
    else
    {
        // a fila será esvaziada
        inicio = null;
        System.out.println("Fila esvaziada");
    }
}
```

```

        while (op != 5);
    }



---



c/c++

```

#include <iostream.h>
#include <conio.h>

//Definindo o registro que representará
//cada elemento da fila

struct FILA
{
    int num;
    FILA *prox;
};

void main()
{
    // a fila está vazia, logo,
    // o ponteiro inicio tem o valor null
    // a operação de remoção acontece no INICIO
    // e a operação de inserção acontece no FIM
    FILA *inicio = NULL;
    FILA *fim = NULL;
    // o ponteiro aux é um ponteiro auxiliar
    FILA *aux;
    // apresentando o menu de opções
    int op;
    do
    {
        clrscr();
        cout<<"\nMENU DE OPÇÕES\n";
        cout<<"\n1 - Inserir na fila";
        cout<<"\n2 - Consultar toda a fila";
        cout<<"\n3 - Remover da fila";
        cout<<"\n4 - Esvaziar a fila";
        cout<<"\n5 - Sair";
        cout<<"\nDigite sua opção: ";
        cin>>op;
        if (op < 1 || op > 5)
            cout<<"Opção inválida!!";
    }
}
```


```

```
if (op == 1)
{
    cout<<"Digite o número a ser inserido na
    ↵ fila: ";
    FILA *novo = new FILA();
    cin>>novo->num;
    novo->prox = NULL;
    if (inicio == NULL)
    {
        // a fila está vazia e o número inserido
        // será o primeiro e o último
        inicio = novo;
        fim = novo;
    }
    else
    {
        fim->prox = novo;
        fim = novo;
    }
    cout<<"Número inserido na fila!!";
}
if (op == 2)
{
    if (inicio == NULL)
    {
        // a fila está vazia
        cout<<"Fila vazia!!";
    }
    else
    {
        // a fila contém elementos
        // e estes serão mostrados
        // do primeiro inserido ao último
        cout<<"\nConsultando toda a fila\n";
        aux = inicio;
        while (aux != NULL)
        {
            cout<<aux->num<<"  ";
            aux = aux->prox;
        }
    }
}
if (op == 3)
{
    if (inicio == NULL)
    {
```

```

        // a fila está vazia
        cout<<"Fila vazia!!";
    }
    else
    {
        // a fila contém elementos
        // e o primeiro elemento inserido
        // será removido
        aux = inicio;
        cout<<"Número "<<inicio->num<<"removido";
        inicio = inicio->prox;
        delete(aux);
    }
}
if (op == 4)
{
    if (inicio == NULL)
    {
        // a fila está vazia
        cout<<"Fila vazia!!";
    }
    else
    {
        // a fila será esvaziada
        aux = inicio;
        while (aux!= NULL)
        {
            inicio = inicio->prox;
            delete(aux);
            aux=inicio;
        }
        cout<<"Fila esvaziada";
    }
}
getch();
}
while (op != 5);
}

```

## Análise da complexidade

A operação de inserção na fila sempre realiza operações básicas, como a de atribuição, para atualizar o INICIO e FIM da fila. O mesmo ocorre no caso da remoção para atualizar o INICIO da fila. Logo, são operações de tempo constante e gastam tempo  $O(1)$ .

Já a operação de consultar toda a fila percorre todos os elementos armazenados nela. Considerando que uma fila contém  $n$  elementos, o tempo de execução será  $O(n)$ .