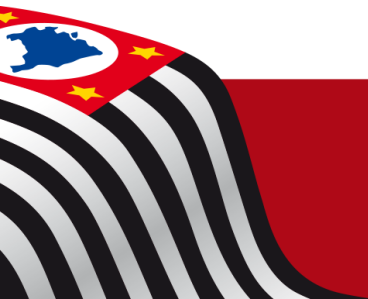


Curso: ADS

Estrutura de Dados

Aula 4 – Filas em C#

Profº Msc. Anderson L. Coan
anderson.coan@fatec.sp.gov.br



Fila - Teoria das Filas

- Uma Fila é um tipo especial de lista linear em que as inserções são realizadas num extremo, ficando as remoções restritas ao outro.
- O extremo onde os elementos são inseridos é denominado final da fila, e aquele de onde são removidos é denominado começo da fila.
- Cada vez que uma operação de inserção é executada, um novo elemento é colocado no final da fila.
- Na remoção, é sempre retornado o elemento que aguarda há mais tempo na fila, ou seja aquele posicionado no começo.
- A ordem de saída corresponde diretamente à ordem de entrada dos elementos na fila de modo que só primeiros elementos que entram são os primeiros a sair.
- Em vista disto, as filas são denominadas listas FIFO (First-In/First-Out).

Filas e Pilhas

conceito

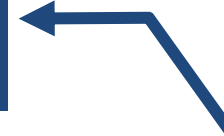
Início da FILA



Primeiro que sai
(FIFO)

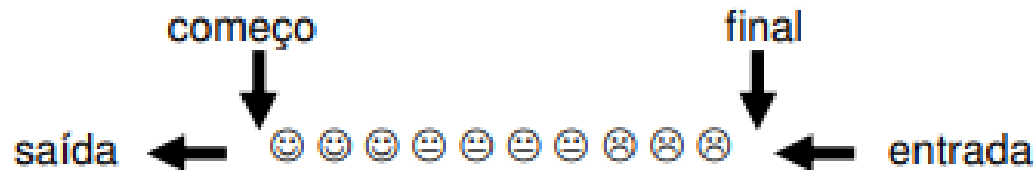


Primeiro
que entra



Fila - Teoria das Filas

- Uma fila funciona como a fila do cinema:
 - a primeira pessoa a entrar na fila é a primeira pessoa a comprar o bilhete.
 - A última pessoa a entrar na fila é a última pessoa a comprar o bilhete (ou - se já estiver esgotado - a não comprar o bilhete).



- Há diversas filas fazendo seu trabalho silenciosamente no sistema operacional de seu computador.
 - Há uma fila de impressão onde seus trabalhos para impressão esperam até a impressora estar disponível.
 - Uma fila também armazena dados do pressionamento de tecla na medida em que você digita no teclado.

Provê modelos para prever o comportamento de sistemas que oferecem serviços para demandas com taxas de chegadas e saídas aleatórias.

Utilizada para modelar sistemas de controle de:

- **Atendimentos**
- **Espera**
- **Saídas**

Exemplos:

- **Sistema telefônico**
- **Sistemas de comunicação de dados**
- **Sistema de impressão**
- **Sistemas de atendimentos em geral**

Tempo de espera de um cliente:

Quanto tempo um cliente espera no banco

Quanto tempo um pacote passa em um roteador

Acúmulo de clientes na fila:

Qual o tamanho médio da fila do banco

Como a fila do roteador se comporta

Tempo ocioso/ocupado dos servidores:

Quanto tempo o caixa fica livre

Qual a utilização do roteador

Taxa de saída (vazão):

Quantos clientes são atendidos por hora

Quantos pacotes são encaminhados por segundo

Funções de manipulação de Filas

- **FUNÇÕES BÁSICAS**

- Seja F uma variável do tipo fila e X um elemento qualquer
enqueue (f, x) - Função que insere X no fim da fila F .

dequeue (f) - Função que remove o elemento do começo da fila F devolvendo o valor do para a rotina que a chamou.

- **FUNÇÕES AUXILIARES**

qinit(f) – Função que esvazia a fila F .

qisfull (f) – Função que retorna um valor lógico informando se a pilha está cheia. Verdadeiro se estiver cheia ou Falso caso contrário.

qisempty (f) – Função que retorna um valor lógico informando se a fila está vazia. Verdadeiro se estiver vazia ou Falso caso contrário.

- Para eliminar o erro lógico, que sinaliza fila vazia e cheia ao mesmo tempo, basta utilizar a implementações de fila circular, onde acrescentamos uma variável contadora para indicar quantos elementos estão armazenados na fila.

Funções de manipulação de Filas

Comando *Queue*

O comando **queue** é utilizado para a criação das filas em C#; sua sintaxe é:

```
Queue<string> minhaFila = new Queue<string>( );
```

onde:

- ❑ <string> = tipo de elemento que será enfileirado;
- ❑ minhaFila = nome da fila;
- ❑ new Queue<string>(); = criação da instância;

Comando *Enqueue*

O comando **Enqueue** é utilizado para adicionar elementos às filas:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Filas
{
    class Program
    {
        static void Main(string[] args)
        {
            Queue<string> minhaFila = new Queue<string>();
            minhaFila.Enqueue("Primeiro Cliente");
            minhaFila.Enqueue("Segundo Cliente");
            minhaFila.Enqueue("Terceiro Cliente");
            minhaFila.Enqueue("Quarto Cliente");
            minhaFila.Enqueue("Quinto Cliente");
        }
    }
}
```


Funções de manipulação de Filas

Comando *Foreach*

Podemos utilizar um comando ***Foreach*** para percorrer uma lista ou objeto. É um dos modos mais fáceis e simples de percorrer uma lista. Uma lista pode ser entendida como uma coleção de dados (vetores, listas, pilhas e listas)

Na declaração do ***foreach***, entre parênteses criamos um elemento do tipo utilizado na coleção e, com o operador ***in***, informamos a coleção a ser percorrida.

Comando ***Foreach*** (Digitar embaixo da fila criada)

```
foreach (string cliente in minhaFila)
{
    Console.WriteLine(cliente);
}

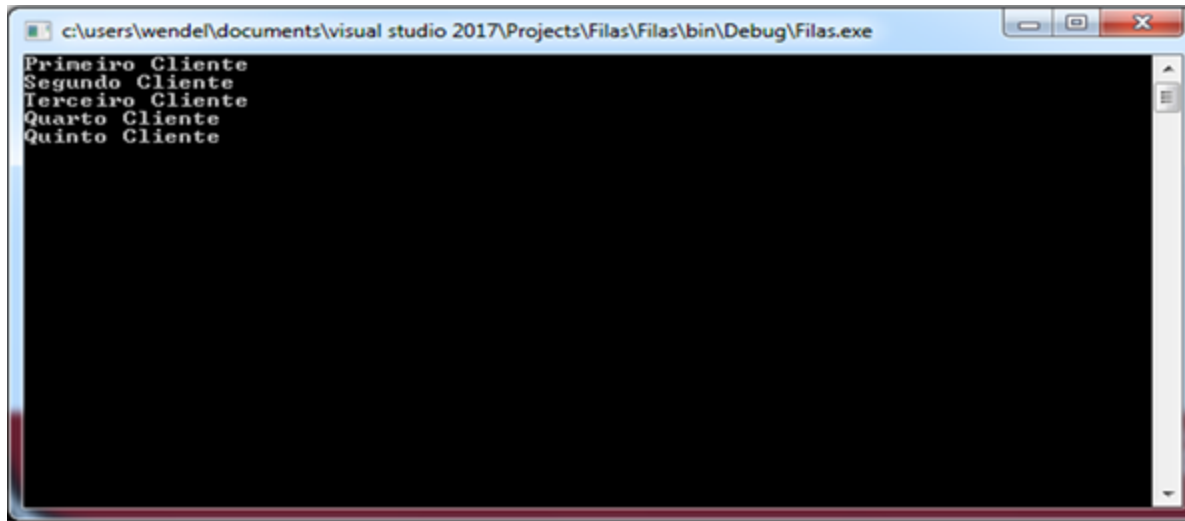
Console.ReadLine();
}
```

Funções de manipulação de Filas

Comando *Foreach*

Fila Criada.

Execute o código para ver sua saída:

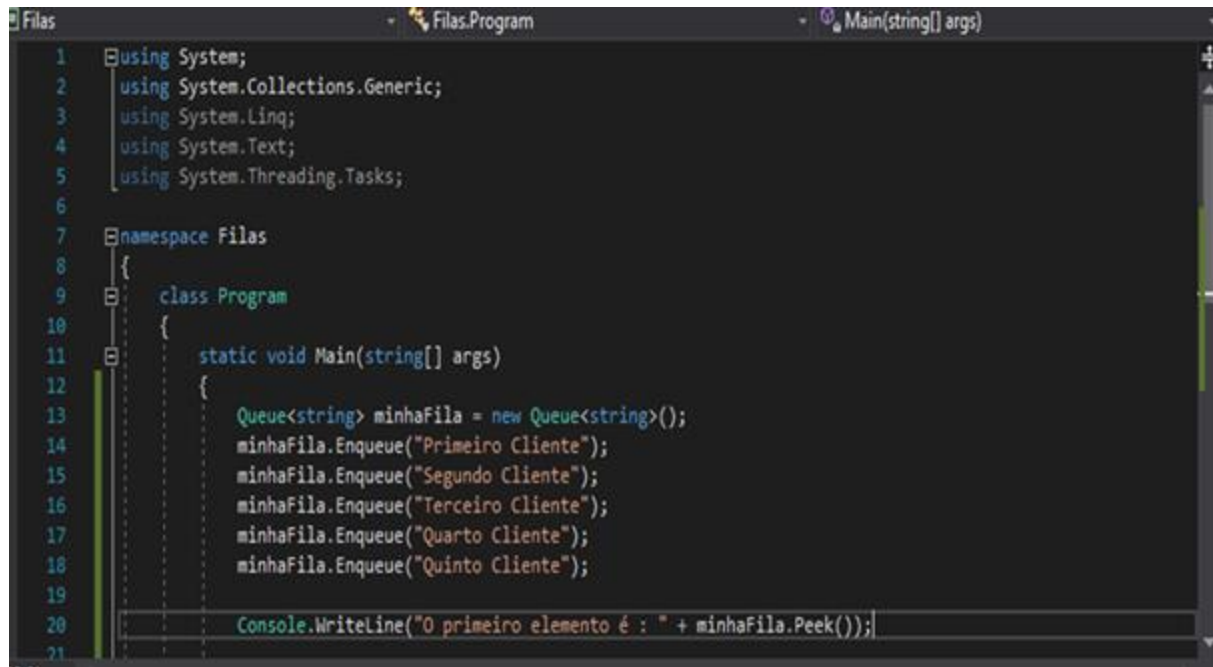


```
c:\users\wendel\documents\visual studio 2017\Projects\Filas\Filas\bin\Debug\Filas.exe
Primeiro Cliente
Segundo Cliente
Terceiro Cliente
Quarto Cliente
Quinto Cliente
```

Funções de manipulação de Filas

Comando *Peek*

Para visualizar o primeiro elemento da fila, usamos o comando ***Peek***:

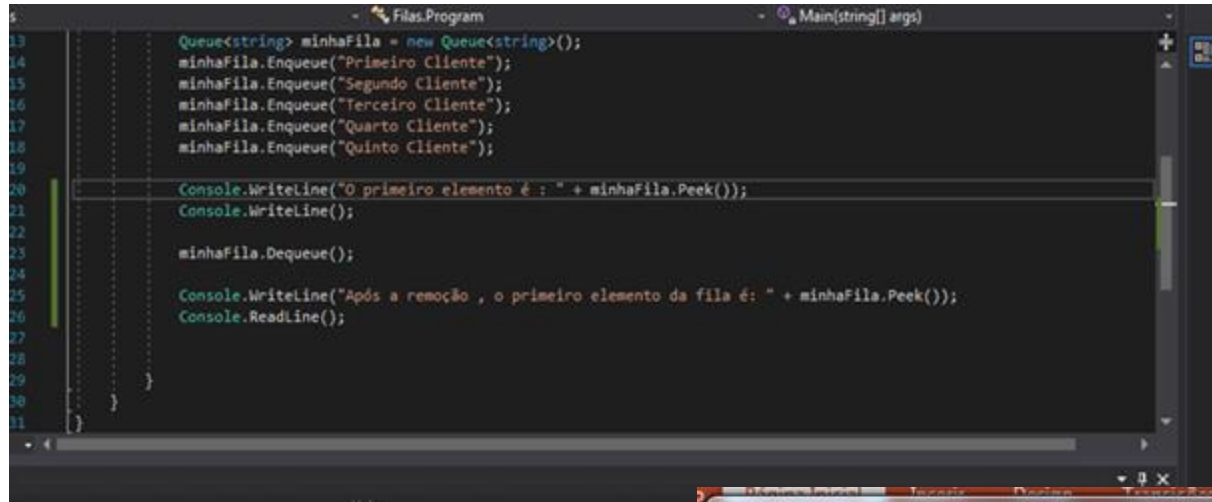


```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Filas
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Queue<string> minhaFila = new Queue<string>();
14             minhaFila.Enqueue("Primeiro Cliente");
15             minhaFila.Enqueue("Segundo Cliente");
16             minhaFila.Enqueue("Terceiro Cliente");
17             minhaFila.Enqueue("Quarto Cliente");
18             minhaFila.Enqueue("Quinto Cliente");
19
20             Console.WriteLine("O primeiro elemento é : " + minhaFila.Peek());
21         }
22     }
23 }
```

Funções de manipulação de Filas

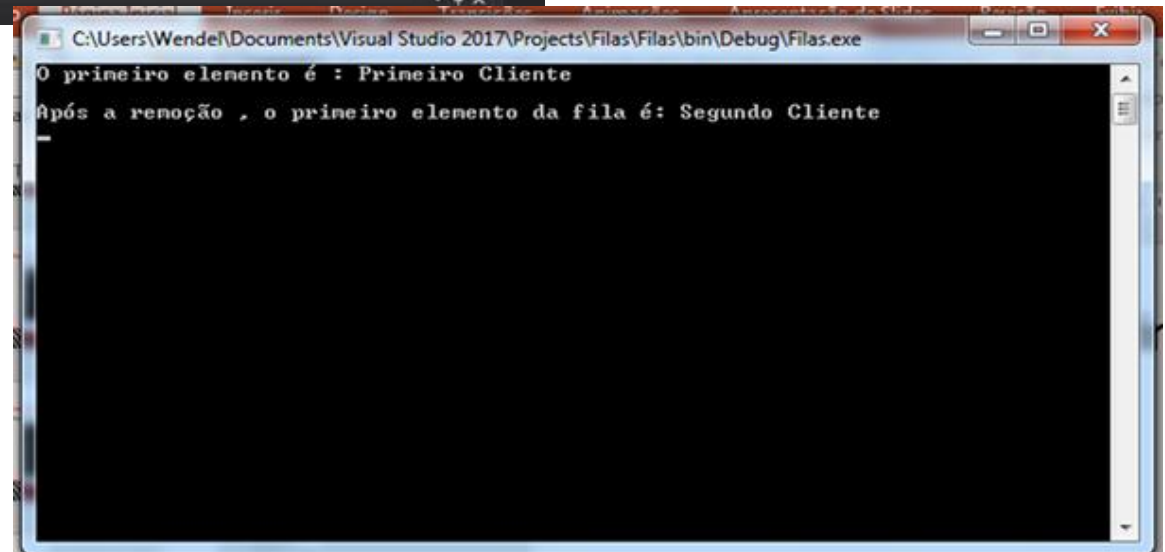
Comando *Dequeue*

Para removermos um elemento da fila, utilizamos este comando, que desenfileira um elemento. É como se o primeiro cliente de uma fila fosse atendido e todos os outros dessem um passo à frente, então o segundo cliente passa a ser o próximo a ser atendido.



```
13 Queue<string> minhaFila = new Queue<string>();
14 minhaFila.Enqueue("Primeiro Cliente");
15 minhaFila.Enqueue("Segundo Cliente");
16 minhaFila.Enqueue("Terceiro Cliente");
17 minhaFila.Enqueue("Quarto Cliente");
18 minhaFila.Enqueue("Quinto Cliente");
19
20 Console.WriteLine("O primeiro elemento é : " + minhaFila.Peek());
21 Console.WriteLine();
22
23 minhaFila.Dequeue();
24
25 Console.WriteLine("Após a remoção , o primeiro elemento da fila é: " + minhaFila.Peek());
26 Console.ReadLine();
27
28 }
29
30 }
```

Saída do comando *Dequeue* →



```
C:\Users\Wende\Documents\Visual Studio 2017\Projects\Filas\Filas\bin\Debug\Filas.exe
O primeiro elemento é : Primeiro Cliente
Após a remoção , o primeiro elemento da fila é: Segundo Cliente
```

Funções de manipulação de Filas

Comando *Count*

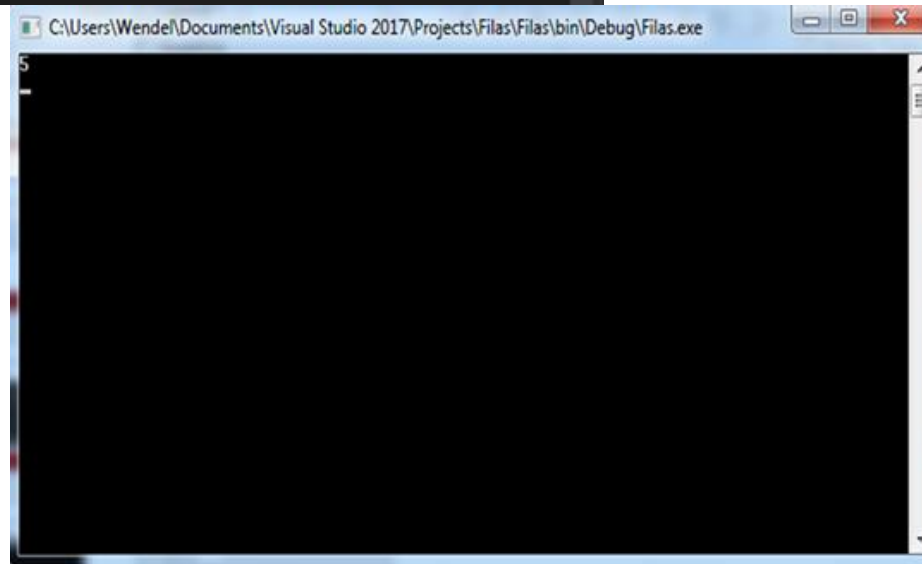
O Count nos permite contar quantos elementos existem numa fila, sua sintaxe é bem simples:

```
class Program
{
    static void Main(string[] args)
    {
        Queue<string> minhaFila = new Queue<string>();
        minhaFila.Enqueue("Primeiro Cliente");
        minhaFila.Enqueue("Segundo Cliente");
        minhaFila.Enqueue("Terceiro Cliente");
        minhaFila.Enqueue("Quarto Cliente");
        minhaFila.Enqueue("Quinto Cliente");

        int contador = minhaFila.Count();

        Console.WriteLine(contador);
        Console.ReadLine();
    }
}
```

Saída do comando *Count* →

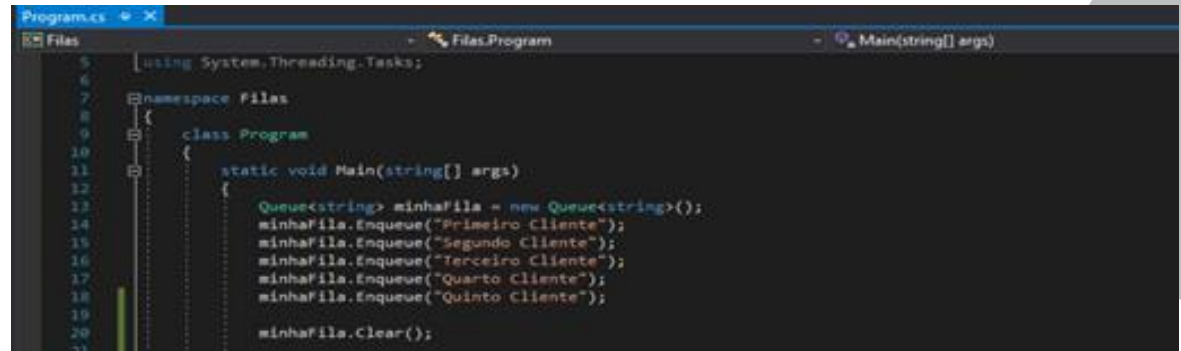


A screenshot of a console window titled "C:\Users\Wende\Documents\Visual Studio 2017\Projects\Filas\Filas\bin\Debug\Filas.exe". The window displays the number "5" on the first line, which is the output of the Count() method from the code above.

Funções de manipulação de Filas

Comando *Clear*

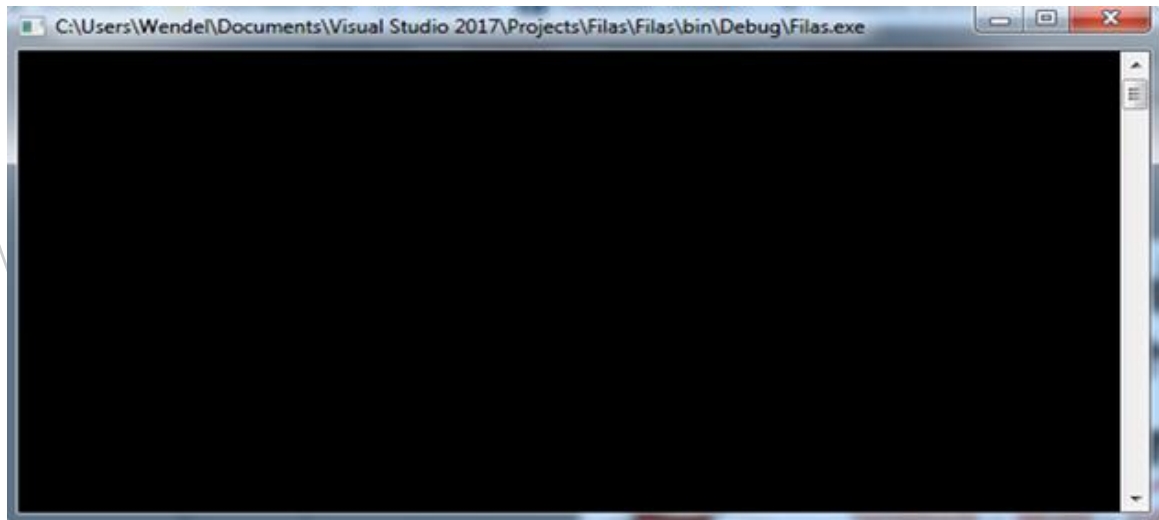
Para limpar nossa fila, usamos o *Clear*:



```
Program.cs + x
Filas
using System.Threading.Tasks;

namespace Filas
{
    class Program
    {
        static void Main(string[] args)
        {
            Queue<string> minhaFila = new Queue<string>();
            minhaFila.Enqueue("Primeiro Cliente");
            minhaFila.Enqueue("Segundo Cliente");
            minhaFila.Enqueue("Terceiro Cliente");
            minhaFila.Enqueue("Quarto Cliente");
            minhaFila.Enqueue("Quinto Cliente");

            minhaFila.Clear();
        }
    }
}
```



← Saída do comando *Clear*

Analizando o código em C: Implementação do arquivo FILAS.H

```
#define Max 50
```

```
typedef int tpElem;
```

```
//Estrutura de dados da fila
```

```
typedef struct
```

```
{
```

```
    int total,comeco,final;
```

```
    tpElem valor[Max];
```

```
}tpFila;
```

```
//protótipo das funções
```

```
void qinit(tpFila *f);      // iniciar a fila
```

```
int qisFull(tpFila *f);    //Verificar se a fila está cheia
```

```
int qisEmpty(tpFila *f);   // verificar se a fila está vazia
```

```
void enqueue(tpFila *f, tpElem x); //colocar um dado no fim da fila
```

```
tpElem dequeue (tpFila *f);      //retirar um dado no começo da fila
```

Analizando o código em C: Implementação do arquivo FILAS.C

```
#include "filas.h"
```

```
int qisFull(tpFila *f)
{
    return (f->total==Max-1);
}
```

```
int qisEmpty(tpFila *f)
{
    return (f->total==0);
}
```

```
void adc (int *i)
{
    (*i)++;
    if(*i==Max)
        *i=0;
}
```

```
void enqueue(tpFila *f, tpElem x)
{
    if (qisFull(f)==0)
    {
        f->valor[f->final]=x;
        adc(&f->final);
        f->total++;
    }
}
```

```
else
{
    printf("\nFila Cheia \n\n");
    system("pause");
}
}
```

```
tpElem dequeue (tpFila *f)
{
    tpElem x;
    if (qisEmpty(f)==0)
    {
        x=(f->valor[f->comeco]);
        adc(&f->comeco);
        f->total--;
        return x;
    }
    else
    {
        printf("\nFila Vazia \n\n");
        system("pause");
    }
}
```


EXERCÍCIOS: FILAS

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

```
int main ( ) {
```

```
tpFila f ;
```

```
char x, y, w, z ;
```

```
qinit ( &f ) ;
```

```
enqueue ( &f, 'a' ) ;
```

```
enqueue ( &f, 'b' ) ;
```

```
printf ( "%c\n", dequeue ( &f ) );
```

```
enqueue ( &f, 'c' ) ;
```

```
enqueue ( &f, 'd' ) ;
```

```
x = dequeue ( &f ) ;
```

```
y = dequeue ( &f ) ;
```

```
enqueue ( &f, 'e' ) ;
```

```
w = dequeue ( &f ) ;
```

```
enqueue ( &f, dequeue ( &f ) ) ;
```

```
printf ( "%c\n", dequeue ( &f ) ) ;
```

```
enqueue ( &f, 'f' ) ;
```

```
z = dequeue ( &f ) ;
```

```
enqueue ( &f, 'a' ) :
```

```
printf ( "%c\n", dequeue ( &f ) );
```

1. Faça a simulação de uma fila F, inicialmente vazia, após a execução de cada um dos comandos do programa abaixo e mostre os valores impressos ao final.

```
printf ( "%c\n", x );
printf ( "%c\n", y );
printf ( "%c\n", w );
printf ( "%c\n", z );
printf ( "\n\n\n" );
system ( "pause" );
return 0;
```

3

[illegible]

Exercícios

1. Crie um sistema de senha de atendimento.

O sistema deve considerar:

- A) Atendimento normal (Inserir um novo elemento, quando ocorrer atendimento, remoção do elemento na fila);
- B) Atendimento prioritário (uma identificação no atendimento para diferenciar do atendimento normal)

Uma sugestão de menu:

- 1-Gerar nova senha
- 2-Efetuar atendimento
- 3-Generar atendimento prioritário
- 4-Efetuar atendimento prioritário
- 6-Exibir a fila atualmente
- 7-Limpar a Fila
- 8-Sair

DÚVIDAS



Curso: ADS Estrutura de Dados Aula 4 – Filas em C#

Profº Msc. Anderson L. Coan
anderson.coan@fatec.sp.gov.br

