Rua José Bongiovani, 700 - CEP 19050-920 - Bloco H - 1º andar - Presidente Prudente SP - (18) 3229-1060 - fipp@fipp.unoeste.br

Estruturas de Dados com disciplina de acesso do tipo FILA

Conceito:

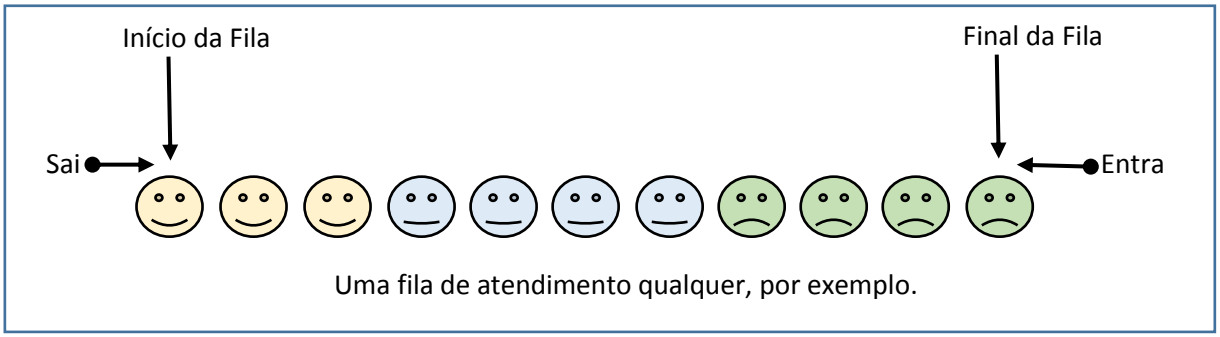
"**FIFO**" ("**F**irst **I**n, **F**irst **O**ut"), ou seja, o "Primeiro a Entrar, será o Primeiro a Sair".

- Todo e qualquer acesso deve ocorrer, exclusivamente, nas duas extremidades da FILA, onde:
  - INÍCIO ➔ somente Retiradas;
  - FIM ➔ somente Inserções.

- Na primeira abordagem de uso de Filas, manteve-se o INÍCIO fixo, portanto, não foi necessário com controlador para ele, já que estaria na posição zero da estrutura de armazenamento durante toda utilização da Fila;
- Para melhorar essa abordagem, eliminando os deslocamentos durante a retirada de um elemento, já que o do Início precisa sair e os demais precisam ser remanejados em direção a posição zero da estrutura, adota-se o controlador INÍCIO, que deve ser incrementado a cada retirada;
- Portanto, no exemplo abaixo, de utilização com **INÍCIO variável**, tem-se a seguinte Estrutura de Dados (ED's):

```
#define MAXFILA 10

struct TpFila
{
        int INICIO, FIM;   ← o controle do Início da Fila se faz necessário
        char FILA[MAXFILA];
};
```



Uma fila de atendimento qualquer, por exemplo.

Mantendo a mesma ideia de implementação como um TAD (Tipo Abstrato de Dados), adotam-se as operações associadas abaixo, como as funções que poderão manipular o conteúdo da Fila proposta.

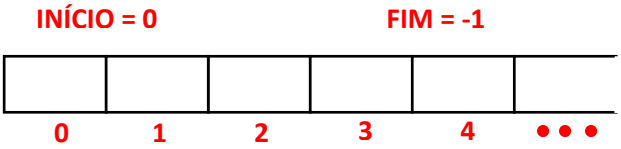
Operações Associadas

- → Inicializar a estrutura para uso;
- → Inserir elemento na Fila;
- → Retirar o elemento do Início da Fila, obrigatoriamente, e o retorna;
- → Retornar elemento que se encontra no INÍCIO da Fila;
- → Retornar elemento que se encontra ao FIM da Fila;
- → Verificar se a FILA está cheia;
- → Verificar se a FILA está vazia;
- → Exibir a Fila.

Implementação em C e exemplo

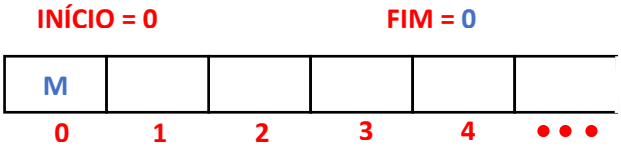
//Obrigatório SEMPRE

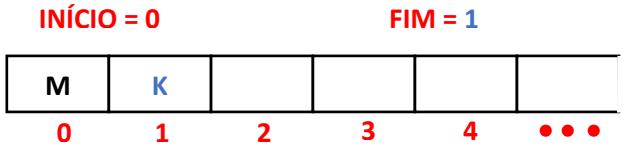
```
void Inicializar (TpFila &F)
{
        F.INICIO = 0;
        F.FIM = -1;
}
```

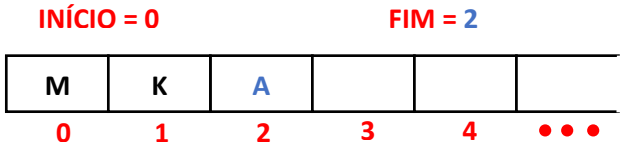


// Inserir M, K e A.

```
void Inserir(TpFila &F, char Elem )
{
        F.FILA[++F.FIM] = Elem;
}
```



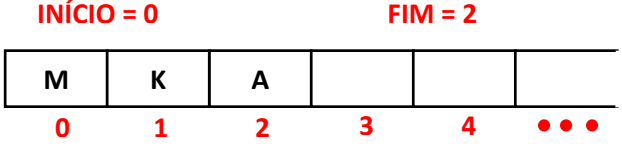




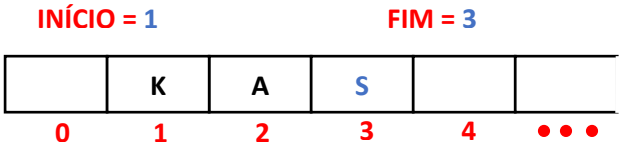
// Retirar, OBRIGATORIAMENTE o elemento do INÍCIO, que deve ser retornado e o INÍCIO incrementado.

Portanto, o **M** deve ser retirado e retornado**.**

```
char Retirar(TpFila &F)
{
        return F.FILA[F.INICIO++];
}
```

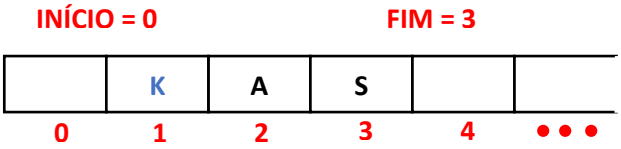


// Inserindo o **S** na Fila, tem-se:



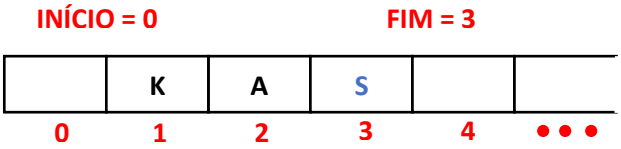
// Retornar o elemento do INÍCIO

```
char ElementoInicio(TpFila F)
{
        return F.FILA[F.INICIO];
}
```



// Retorna o elemento do FIM

```
char ElementoFim(TpFila F)
{
        return F.FILA[F.FIM];
}
```



// Verificar se a FILA está cheia

```
int Cheia(int inicio, int fim)
{
        return ( inicio == 0 && fim == MAXFILA -1 );
}
```

→ **Será que é o ideal???**

// Verificar se a FILA está vazia

```
int Vazia(int inicio, int fim)
{
        return(inicio > fim);
}
```

→ **Será que é o ideal? Resolve todos os casos?**

// Exibir a FILA

```
void Exibir(TpFila F)
{
        while (!Vazia(F.INICIO,F.FIM))
                printf("\t %c",Retirar(F));
}
```

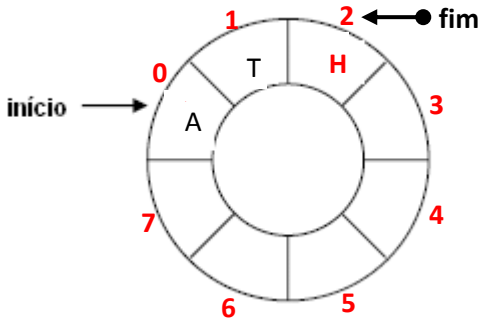
FILA CIRCULAR

O conceito continua o mesmo das abordagens anteriores, com o objetivo de reaproveitamento total dos espaços, que por ventura, deixarão de ter novas utilizações e, claro, eliminando toda e qualquer possibilidade de deslocamentos na estrutura de armazenamento.

Conceito:

"**FIFO**" ("**F**irst **I**n, **F**irst **O**ut"), ou seja, o "Primeiro a Entrar, será o Primeiro a Sair".

- Todo e qualquer acesso deve ocorrer, exclusivamente, pelos controladores da FILA, onde:
  - INÍCIO ➔ somente Retiradas;
  - FIM ➔ somente Inserções.
- Considerando que a estrutura utilizada se torna algo contínuo, ou seja, uma estrutura circular.



Por essa razão, a Estrutura de Dados e algumas operações precisam ser repensadas para que a implementação das mesmas reflita a otimização de acessos e utilização dos espaços disponíveis.

Portanto, tem-se a seguinte ED's:

```
#define MAXFILA 10

struct TpFilaCirc
{
        int INICIO, FIM, CONT;
        char FILA[MAXFILA];
};
```

O controle da quantidade (CONT) de elementos facilita identificar se a Fila está Cheia ou Vazia, caso contrário, são necessárias várias considerações para estabelecer uma dessas condições. Ou, alterar toda linha de resolução, tendo de utilizar operações de ponto flutuante. Observar essa abordagem no livro de Nivio Ziviani (Projeto de Algoritmos).

Operações Associadas

→ Inicializar a estrutura para uso;
→ Inserir elemento na Fila;

- → Retirar o elemento do Início da Fila, obrigatoriamente, e o retorna;
- → Retornar elemento que se encontra no INÍCIO da Fila;
- → Retornar elemento que se encontra ao FIM da Fila;
- → Verificar se a FILA está cheia;
- → Verificar se a FILA está vazia;
- → Exibir a Fila.

Implementação em C e exemplo

```
//Protótipos das operações associadas...

void FCInicializar(TpFilaCirc &F);
void FCInserir(TpFilaCirc &F, char Elem);
char FCRetirar ( TpFilaCirc &F);
char FCElementoInicio(TpFilaCir F);
char FCElementoFIM(TpFilaCir f )
int FCVazia(int cont);
int FCCheia(int cont);
void FCExibir(TpFilaCirc F);
```