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 COM PRIORIDADES

Conceito: FIFO, mas dentro da mesma prioridade.

O elemento deve ser inserido ao **final da FILA** e deslocado de forma a manter a ordem vinculada a prioridade de execução/atendimento dos elementos. Por exemplo, se dois itens na fila têm a mesma prioridade, então aquele que entrou primeiro sairá primeiro, porém se eles têm prioridades distintas, então aquele com maior prioridade sairá primeiro.

Porém, as Estruturas de Dados precisam contemplar os detalhes dos elementos da Fila com as devidas Prioridades. Já as operações associadas são as mesmas de uma FILA qualquer, havendo apenas mudança na operação de inserção, pois é preciso realizar o deslocamento do *Elemento* inserido até o local adequado, conforme a sua *Prioridade*.

Dessa forma, pode-se definir as ED's como:

```
#define MAXFILA 10

struct TpElemento
{
        char Elemento;
        int Prioridade;
};

struct TpFilaPrioridade
{
        int INICIO, FIM, CONT;
        TpElemento FILA[MAXFILA];
};
```

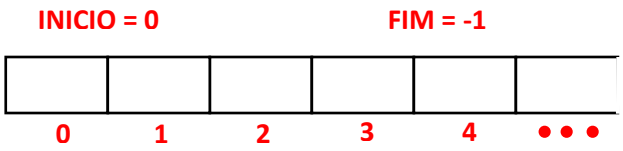
Operações Associadas

→ Inicializar a estrutura para uso;
→ Inserir elemento na Fila; ➔ **deve contemplar a organização do elemento ao inseri-lo, ou seja, o uso do Método de Ordenação Inserção Direta (*Insection Sort*);**
→ 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 e Exemplo com Prioridades: [1] Alta - [2] Média - [3] Baixa

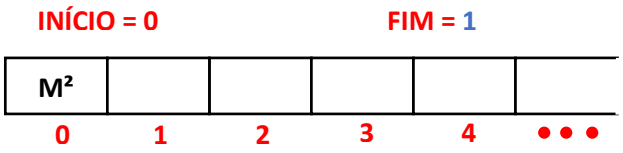
//Obrigatório SEMPRE

```
void FPInicializar(TpFilaPrioridade &FP)
{
        FP.INICIO = 0;
        FP.FIM = -1;
        FP.CONT = 0;
}
```

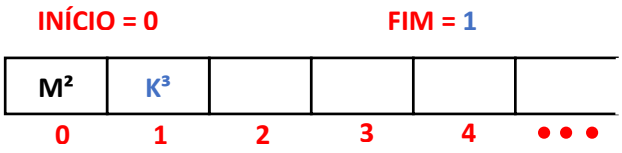



```
void FPInserir(TpFilaPrioridade &FP, TpElemento Elemento)
{
        TpElemento aux;
        int i;
        FP.FILA[++FP.FIM] = Elem;
        i = FP.FIM;          FP.CONT++;
        while(i > FP.INICIO && FP.FILA[i].Prioridade < FP.FILA[i-1].Prioridade )
        {
                aux = FP.FILA[i];
                FP.FILA[i] = FP.FILA[i-1];
                FP.FILA[i-1] = aux;
                i--;
        }
}
```

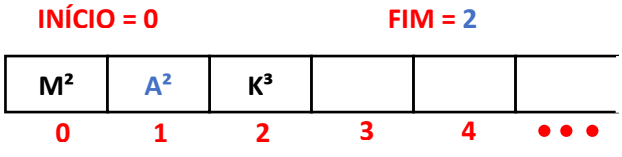
// Insere M com prioridade 2.



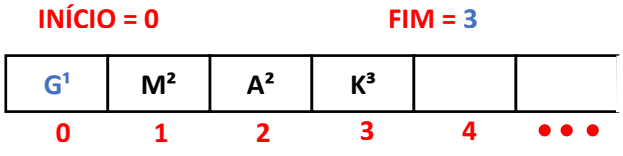
// Insere K com prioridade 3.



// Insere A prioridade 2.



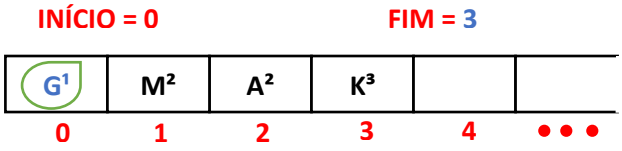
// Insere G prioridade 1.



//Retirar, OBRIGATORIAMENTE o elemento que se encontra no INÍCIO da FILA, independente da Prioridade. E, por fim, deve ser retornado o Elemento, neste caso um *TpElemento*.

Portanto, o G^1 deve ser retirado e retornado.

```
TpElemento FPRetirar(TpFilaPrioridade &FP)
{       FP.CONT--;
        return FP.FILA[FP.INICIO++];
}
```



// Retornar o elemento do INÍCIO

```
TpElemento FPElementoInicio(TpFilaPrioridade FP)
{
        return FP.FILA[FP.INICIO];
}
```

//Retornar o elemento do FIM

```
TpElemento FPElementoFim(TpFilaPrioridade FP)
{
        return FP.FILA[FP.FIM];
}
```

// Verificar se a FILA está Cheia

```
char FPCheia(int cont)
{
        return (cont == MAXFILA);
}
```

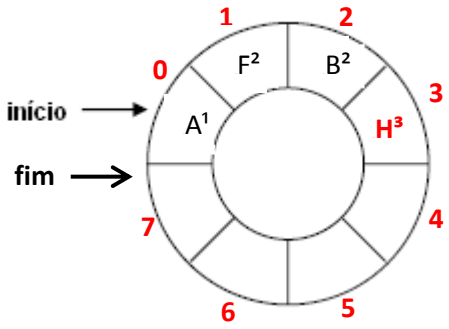
//Verificar se a FILA está Vazia

```
char FPVazia(int cont)
{
        return (cont == 0);
}
```

// Exibir a FILA com PRIORIDADES

```
void FPExibir(TpFilaPrioridade FP)
{
  TpElemento Aux;
  while (!Vazia(FP.CONT))
  {
    Aux = FPRetirar(FP);
    printf("\nElemento: %c – Prioridade: %d", Aux.Elemento, Aux.Prioridade);
  }
}
```

FILA CIRCULAR COM PRIORIDADES



O conceito continua o mesmo do que FILA COM PRIORIDADE, sendo assim o elemento é inserido no final e deslocado conforme sua prioridade.

As Estruturas de Dados e as Operações Associadas devem ser adaptadas em relação à FILA CIRCULAR sem prioridades e o INSERIR deve ser remodelado.

Portanto, têm-se as seguintes Estruturas de Dados:

```
#define MAXFILA 10

struct TpElemento
{
        char Elemento;
        int Prioridade;
};

struct TpFilaCircPrioridade
{
        int INICIO, FIM, CONT;
        TpElemento FILA[MAXFILA];
};
```

Operações Associadas

→ Inicializar a estrutura para uso;
→ Inserir elemento na Fila; ➔ **deve contemplar a organização do elemento ao inseri-lo, ou seja, o uso do Método de Ordenação Inserção Direta (*Insection Sort*) com a questão circular contida na solução;**
→ 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.

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

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