2. FILA COM PRIORIDADES em C - Estrutura de Dados

2.1 Conceito

- Mesmo conceito da FILA, se baseia no **FIFO**("First In, First Out"), no entanto, dentro da mesma prioridade.
- Uma **fila com prioridades** é uma estrutura de dados com duas operações básicas:
  - Inserir um novo elemento;
  - Remover o elemento com maior prioridade.
- É uma estrutura de dados que mantém uma coleção de elementos, cada um com uma prioridade associada.
- 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.
- Um exemplo é se dois itens na fila têm a mesma prioridade , então aquele que entrou primeiro sairá primeiro, porém se eles tem prioridades distintas, então aquele com maior prioridade sairá primeiro.
- As operações associadas são muito semelhantes à 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 sua prioridade.

2.2 Exemplos de Aplicação

- Fila de banco no qual existe prioridade dedicada à gestantes e idosos;
- Fila de uma lotérica;
- Supermercado com caixa dedicado à idosos;
- Fórum;
- Cinema;
- Entre outros.

2.3 Estrutura da FILA COM PRIORIDADES

TAD (Tipo Abstrato de Dados) da FILA COM PRIORIDADES:

```c
#define MAXFILA 10

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

2.4 Operações Associadas

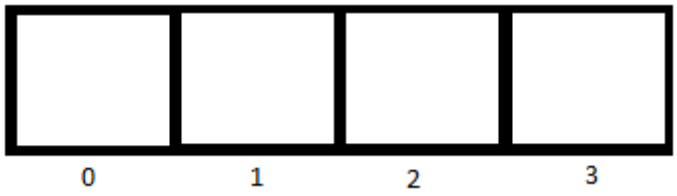
- void FPInicializar (TpFilaPrioridade &FP)
- void FPInserir (TpFilaPrioridade &FP, TpElemento Elemento) -> usa o Método de Ordenação *Insertion Sort*
- TpElemento FPRetirar (TpFilaPrioridade &FP)
- TpElemento FPElementoInicio (TpFilaPrioridade FP)
- TpElemento FPElementoFim (TpFilaPrioridade FP)
- char FPCheia (int cont)
- char FPVazia (int cont)
- void FPExibir (TpFilaPrioridade FP)

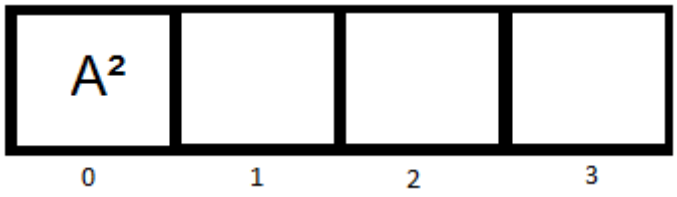
2.5 Implementação em C

```c
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] = Elemento;
    i = FP.FIM;
    while(i > FP.INICIO && FP.FILA[i].Prioridade < FP.FILA[i-1].Prioridade ){
//enquanto o i(elemento final do vetor) for maior que a posicao inicial (ou
seja, vai percorrer todo o vetor) e a prioridade da ultima for maior que a
antipenultima, irá acontecer o remanejamento
        aux = FP.FILA[i];
        FP.FILA[i] = FP.FILA[i-1];
        FP.FILA[i-1] = aux;
        i--;
    }
    FP.CONT ++;
}
TpElemento FPRetirar(TpFilaPrioridade &FP){
    FP.CONT--;
    return FP.FILA[FP.INICIO++];
}
```

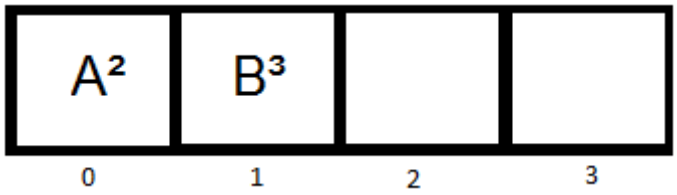
```
TpElemento FPElementoInicio(TpFilaPrioridade FP){
    return FP.FILA[FP.INICIO];
}
TpElemento FPElementoFim(TpFilaPrioridade FP){
    return FP.FILA[FP.FIM];
}

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

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

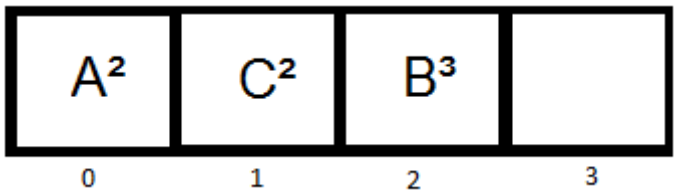
2.6 Funcionamento

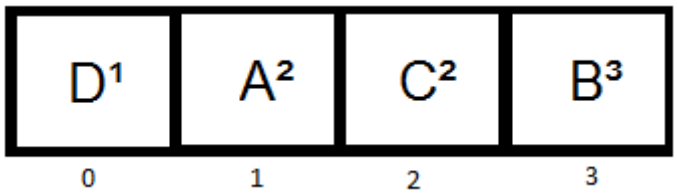
1. Inicializar:
    - Início = 0
    - Fim = -1
    - Contador = 0
    - Fila = 

2. Inserir 'A' com prioridade 2:
    - Início = 0
    - Fim = 0
    - Contador = 1
    - Fila = 

3. Insere 'B' com prioridade 3:
    - Início = 0
    - Fim = 1
    - Contador = 2

- Fila = 

4. Insere 'A' com prioridade 2:
    - Início = 0
    - Fim = 2
    - Contador = 3
    - Fila = 

5. Insere 'D' com prioridade 1:
    - Início = 0
    - Fim = 3
    - Contador = 4
    - Fila = 

6. Retirar (obrigatoriamente retorna o elemento do início da fila, nesse exemplo o TpElemento):
    - Início = 1
    - Fim = 3
    - Contador = 3
    - Fila =