FILA
Algoritmos e Programação II

(slides baseados no material do prof. Fábio Viduani)

INTRODUÇÃO

- Lista linear especial
- Política de inserções e remoções bem definida
- Inserção e remoção são as únicas operações

DEFINIÇÃO

- FILA é uma lista linear com **dois extremos** destacados e tal que as operações de inserção são realizadas em **um** dos extremos da lista e a remoção é realizada no **outro** extremo.

- Funcionamento dessa estrutura pode ser comparado a qualquer fila que usamos com frequência como, por exemplo, uma fila de um Banco.

EXEMPLOS

Exemplos de filas são:

- Fila de caixa de banco
- Fila de vagões de trem

APLICAÇÕES

- Aplicações de FILA:
  - Fila de arquivos para impressão;
  - Atendimento de processos requisitados ao sistema operacional;
  - Buffer para gravação de dados em mídia;
  - Processos de comunicação em redes de computadores

DEFINIÇÃO

- Considere que as células da fila são do tipo abaixo:

C

```
typedef struct cel {
   int chave;
   struct cel *prox;
} celula;
```

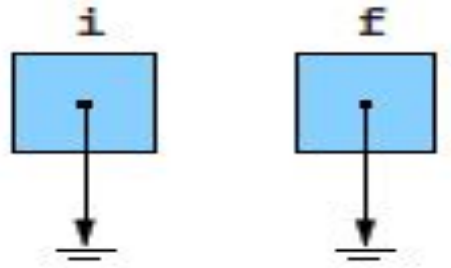
C++

```
struct celula{
   int chave;
   struct celula *prox;
};
```

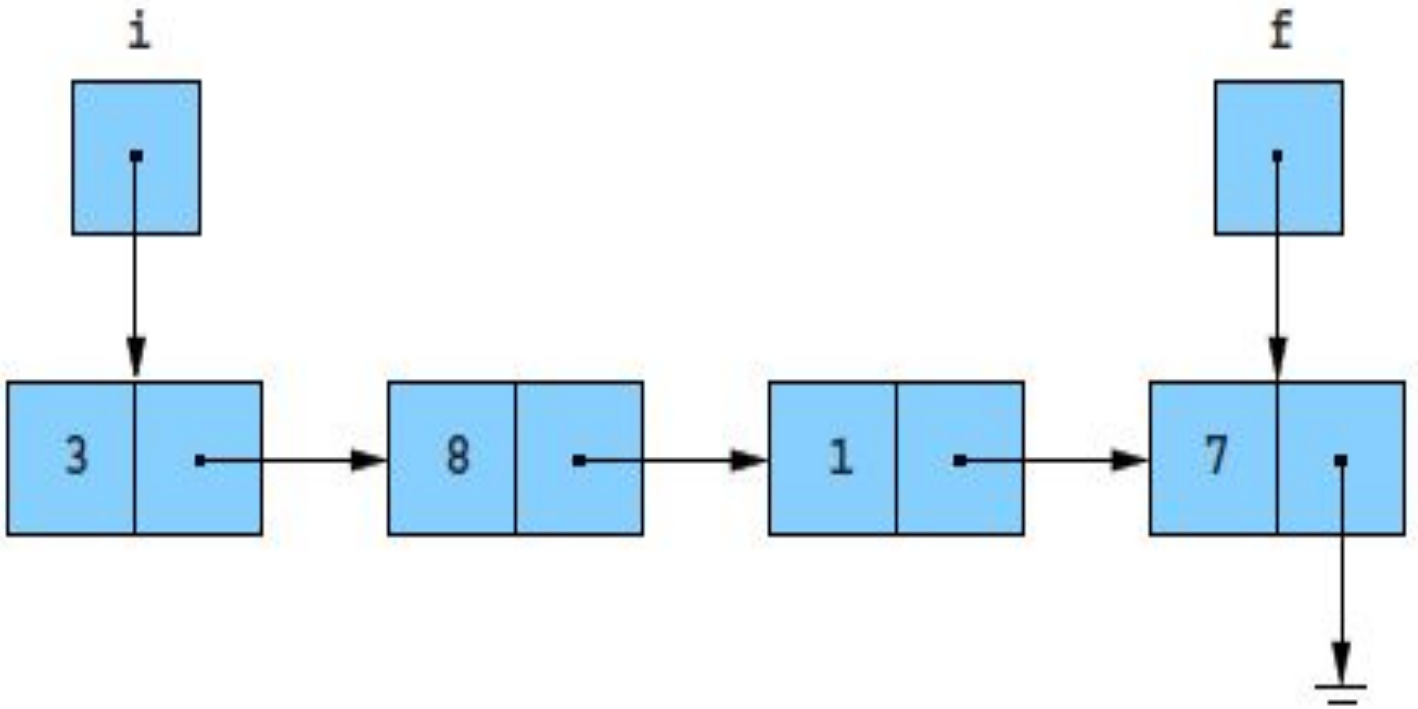
DEFINIÇÃO

- A inicialização de uma fila vazia em alocação encadeada é dada a seguir: FILA 2 extremos

```
celula *i, *f;
i = NULL;
f = NULL;
```



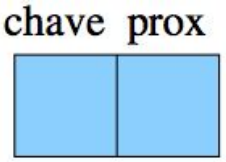
REPRESENTAÇÃO DE UMA **FILA VAZIA** EM ALOCAÇÃO ENCADEADA



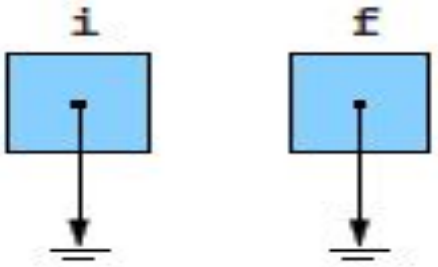
REPRESENTAÇÃO DE UMA **FILA** EM ALOCAÇÃO ENCADEADA

OPERAÇÕES

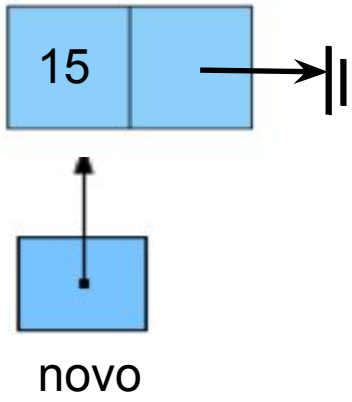
- As operações básicas são:
  - enfileirar
  - desenfileirar



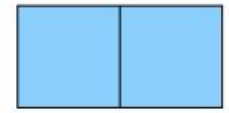
ENFILEIRAR



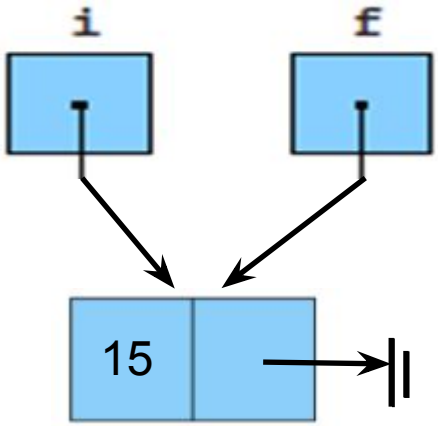
X = 15

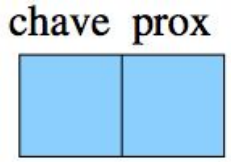


chave prox

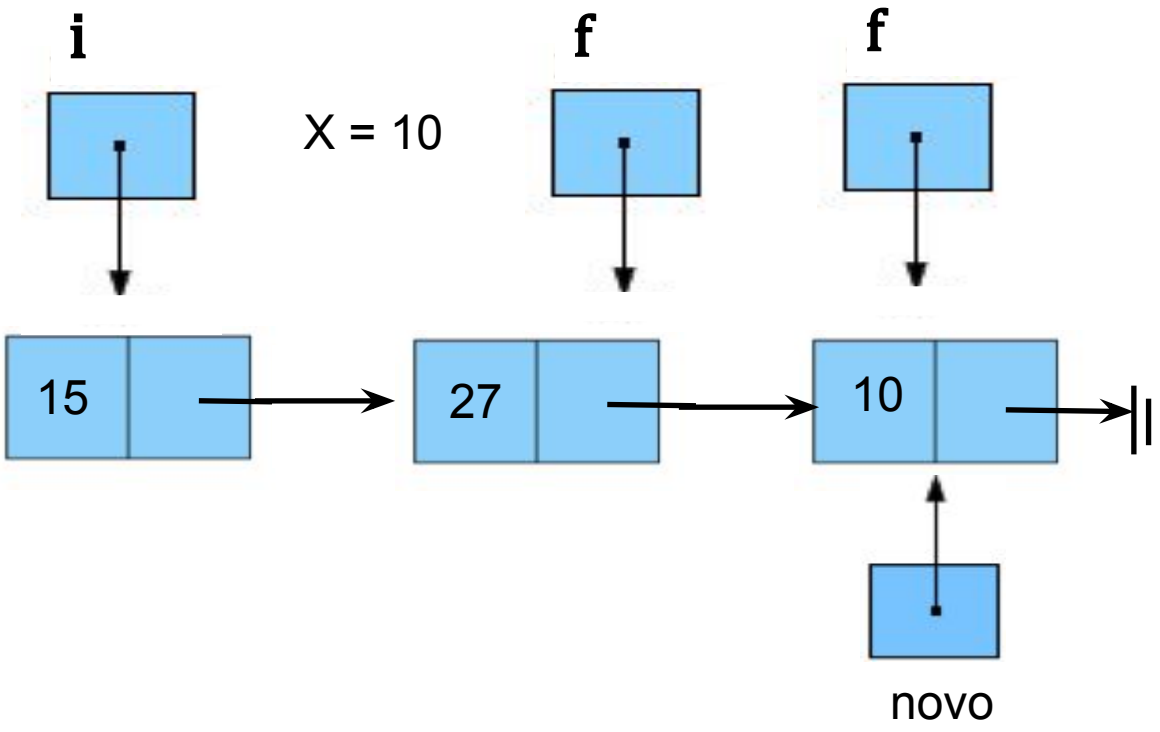


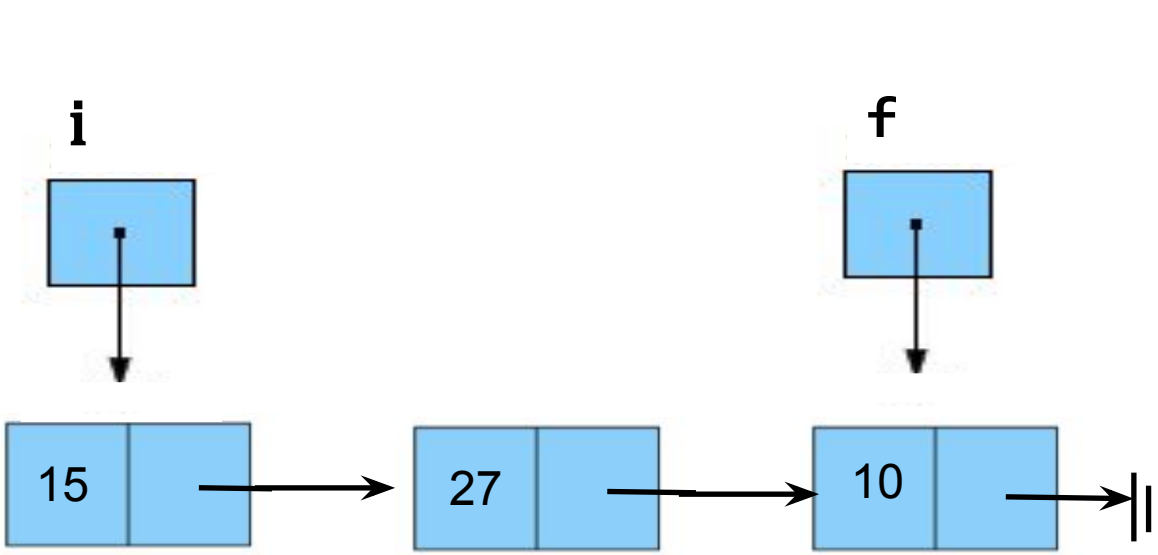
ENFILEIRAR



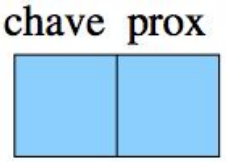


ENFILEIRAR



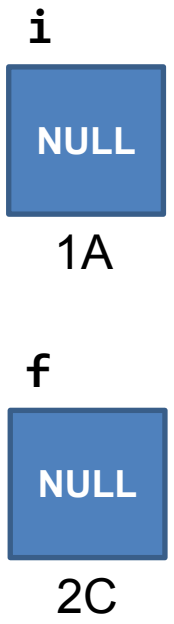


ENFILEIRAR



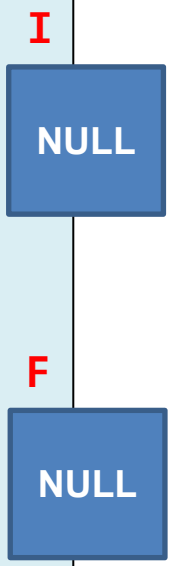
OPERAÇÃO: enfileirar

```
/*MAIN*/

int main()
{
    celula *i = NULL, *f = NULL; //criando uma fila vazia
    int n;
…
    scanf(“%d”, &n);
    while( n!= 0 ){
        enfileirar(   n,   i,   f );
        scanf(“%d”, &n);

    }
…
}
```



OPERAÇÃO: enfileirar

```
/*Função recebe um ponteiro para o início da fila, um ponteiro
para o final da fila e o elemento x que deseja enfileirar*/
void enfileirar(int x, celula *&I, celula *&F)
{
    celula *nova;

    nova = (celula *) malloc(sizeof (celula));
    nova->chave = x;
    nova->prox = NULL;

    if(I == NULL)
        I = F = nova;
    else{
       F->prox = nova;
       F = nova;
    }
}
```



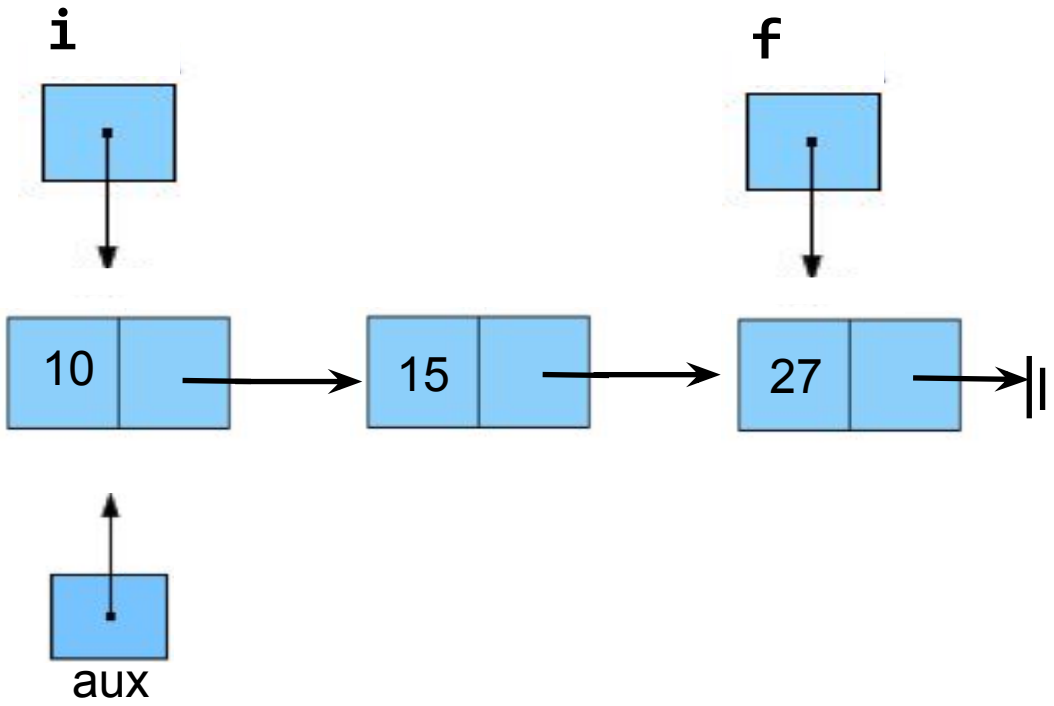
I é um apelido p/ i

F é um apelido p/ f

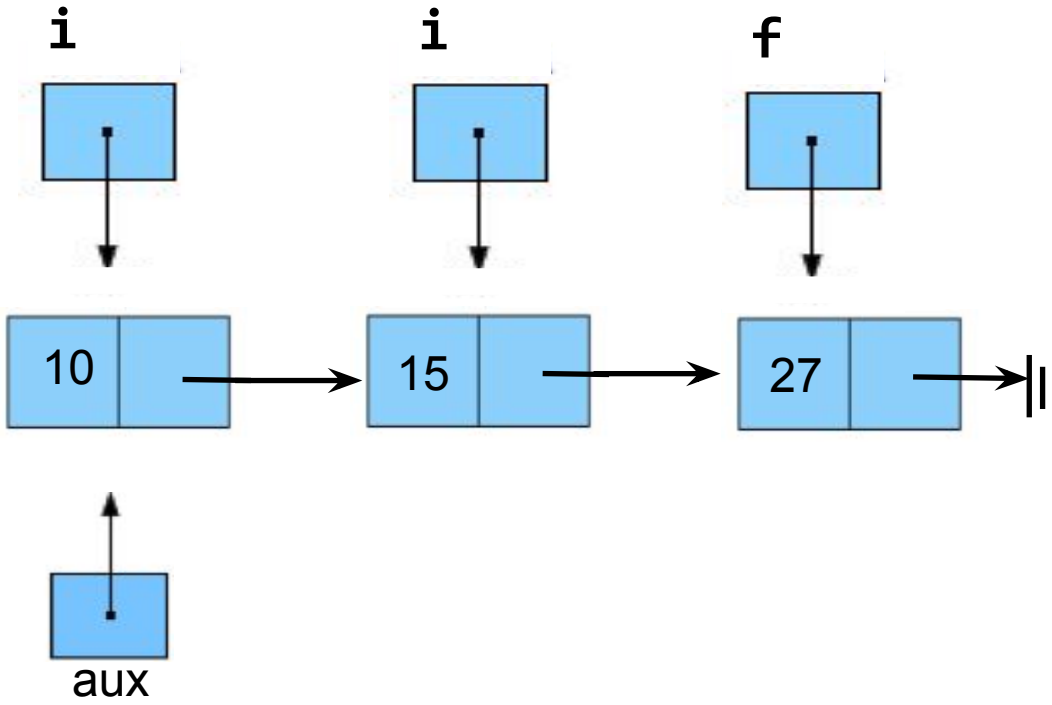
TEMPO DE EXECUÇÃO → T(n) = O(1) - tempo constante

OPERAÇÃO: desenfileirar

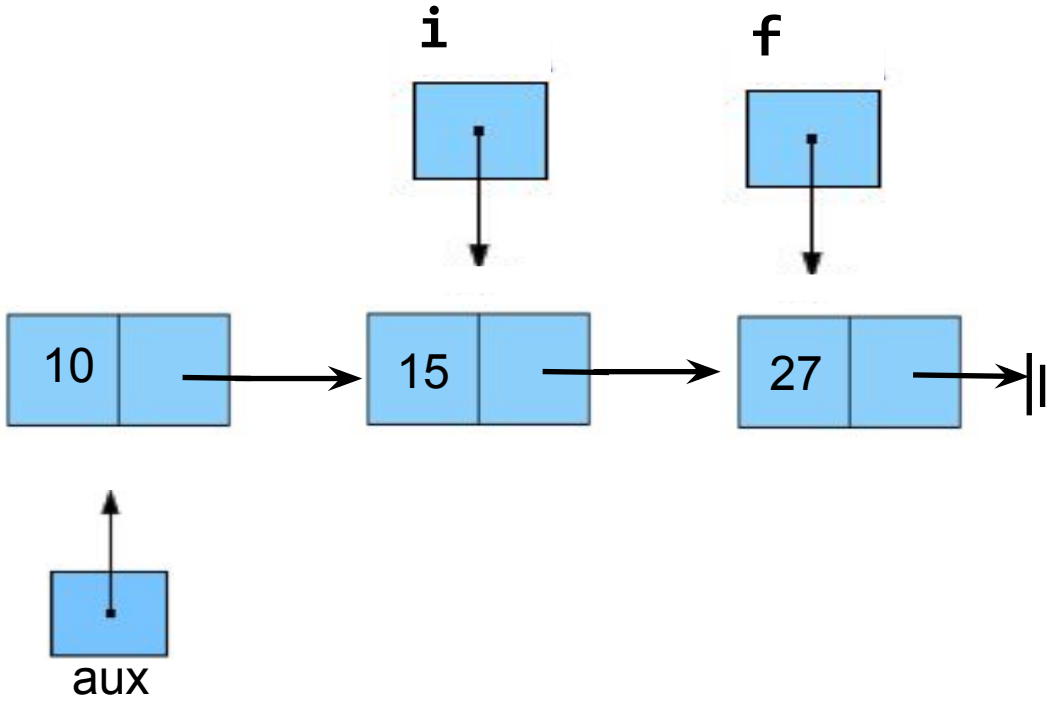
CASO 1: quantidade de NÓS > 1



OPERAÇÃO: desenfileirar

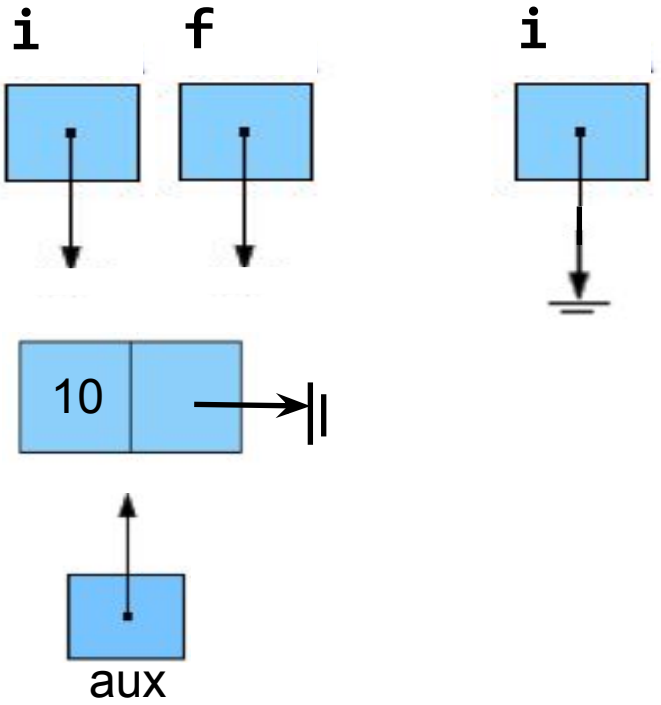


OPERAÇÃO: desenfileirar



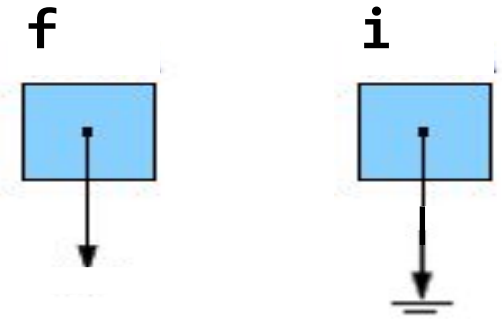
OPERAÇÃO: desenfileirar

CASO 2: quantidade de NÓS = 1



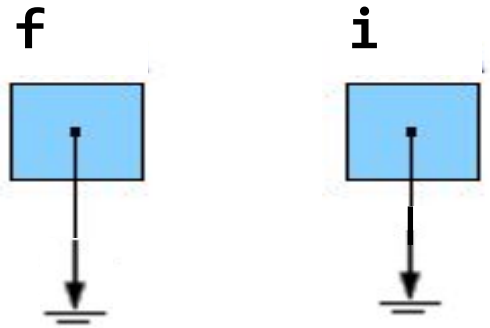
OPERAÇÃO: desenfileirar

CASO 2: quantidade de NÓS = 1



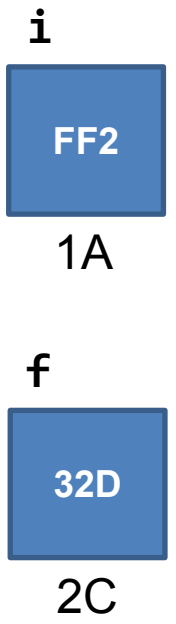
OPERAÇÃO: desenfileirar

CASO 2: quantidade de NÓS = 1



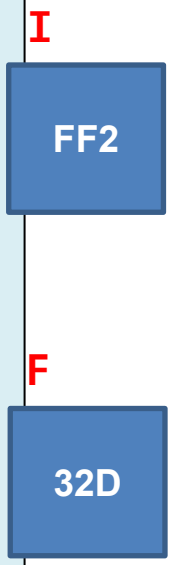
OPERAÇÃO: desenfileirar

```
/*MAIN*/

int main()
{
    celula *i = NULL, *f = NULL; //criando uma fila vazia
    int n;
…
…

    n = desenfileirar(   i,   f );


…
}
```



OPERAÇÃO: desenfileirar

```
/*Função recebe os ponteiros dos extremos da fila,
desenfileira e retorna o valor desenfileirado*/
int desenfileirar(celula *&I, celula *&F)
{
    celula *aux;
    int num;

    if(I == NULL)
        return 0;
    else{
        num = I->chave;
        aux = I;
        I = aux->prox;
        if(I == NULL)
            F = NULL;
        free(aux);
        return num;
    }
}
```



TEMPO DE EXECUÇÃO → T(n) = O(1) - tempo constante