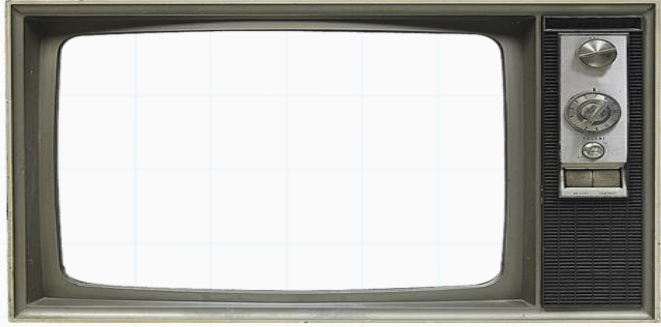
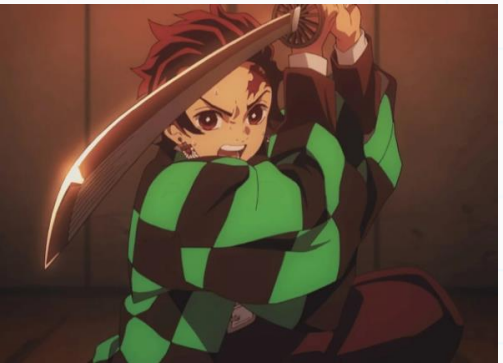
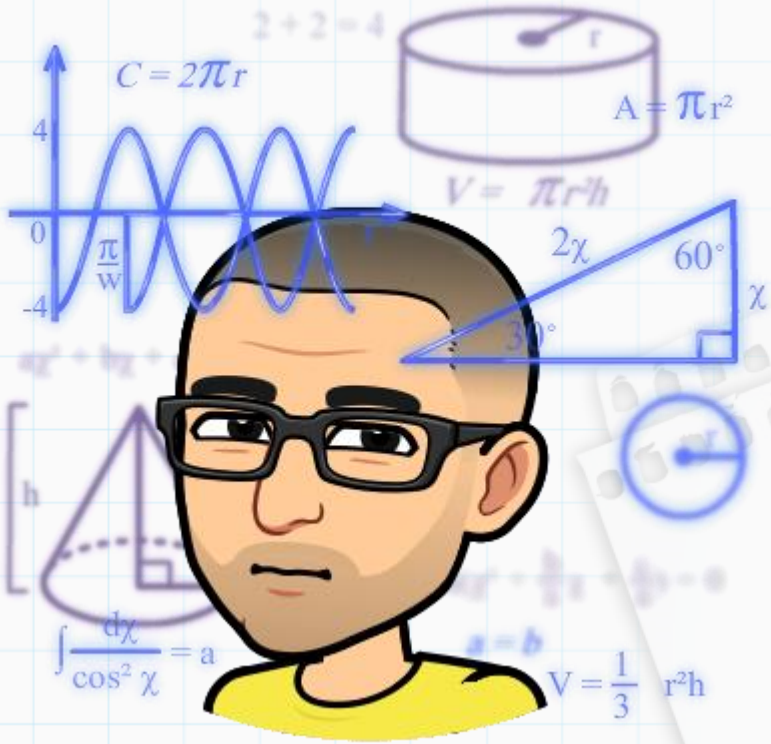
ProgramaçãoEstruturada



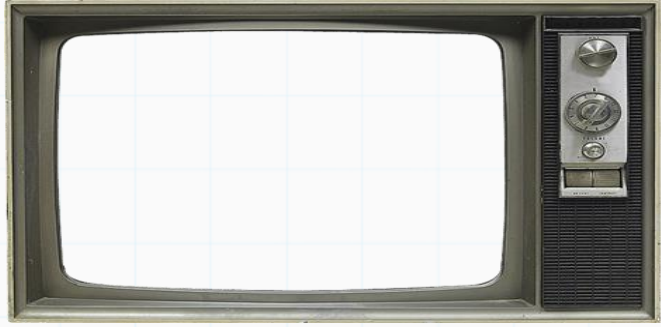
Professor : Yuri Frota

yuri@ic.uff.br





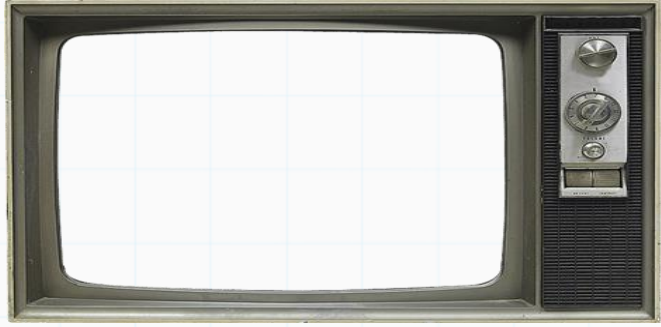
Filas



- Filas são listas com acessos com operações especiais (acessos simplificados e restritos)

- São estruturas do tipo FIFO (First in – first out)

Filas



- Filas são listas com acessos com operações especiais (acessos simplificados e restritos)

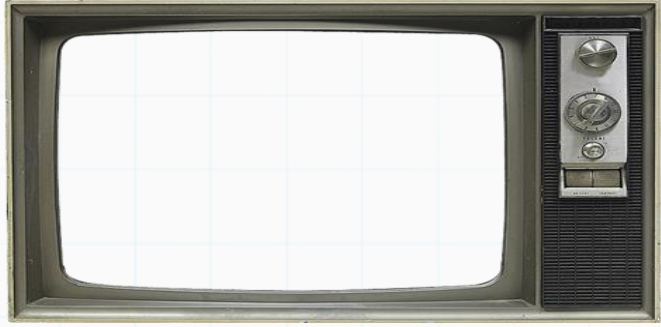
- São estruturas do tipo FIFO (First in – first out)

Sai da fila

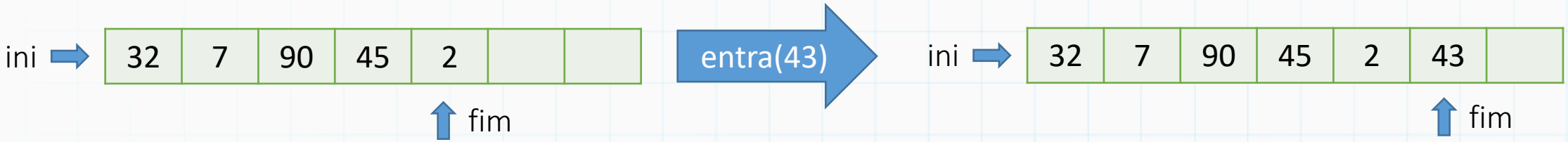
Entra na fila

se tirar ou colocar no meio vão reclamar que está furando a fila

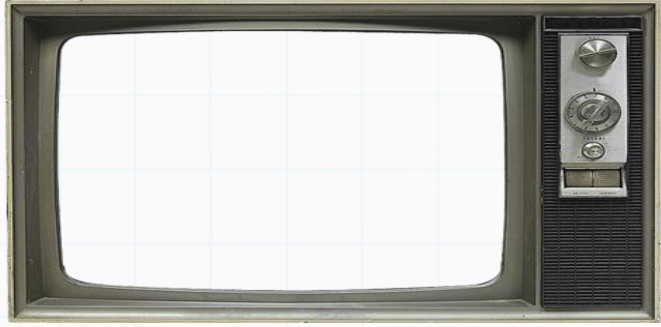
Filas



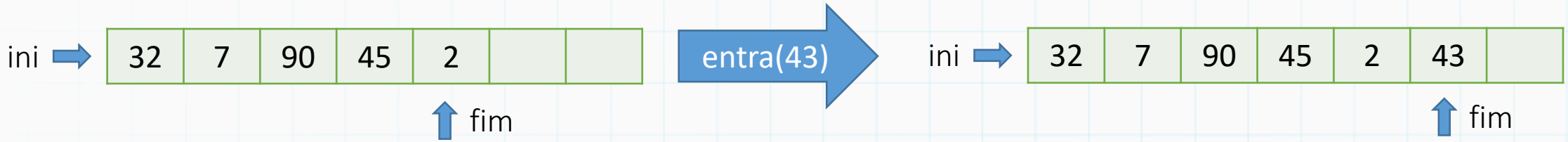
- TAD de Filas :
  - Dados:
    - Elementos (são os dados)
    - Ponteiro para o início e para o fim da fila
  - Operações:
    - entra(F, x): insere um elemento x no fim da fila (enfileira).

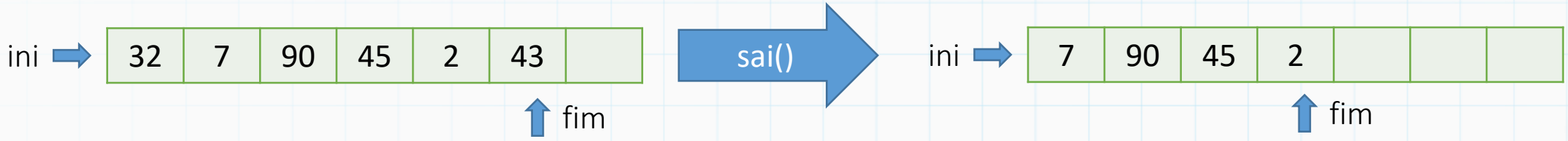


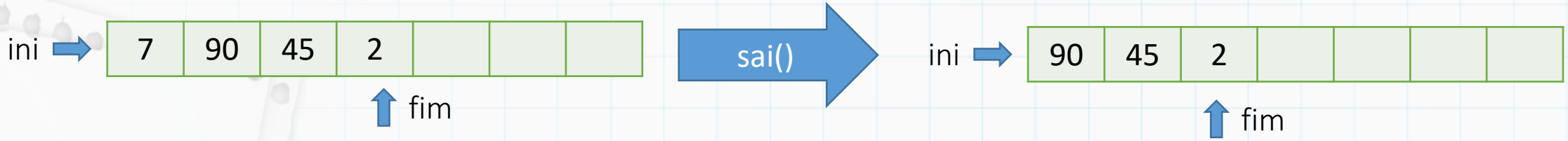
Filas



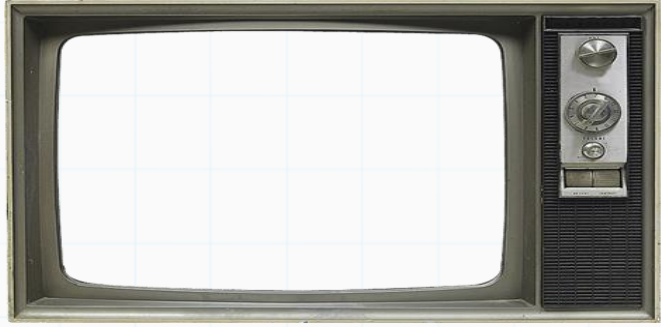
- TAD de Filas :
  - Dados:
    - Elementos (são os dados)
    - Ponteiro para o início e para o fim da fila
  - Operações:
    - entra(F, x): insere um elemento x no fim da fila (enfileira).
    - sai(F): retira o elemento no começo da fila (desenfileira).





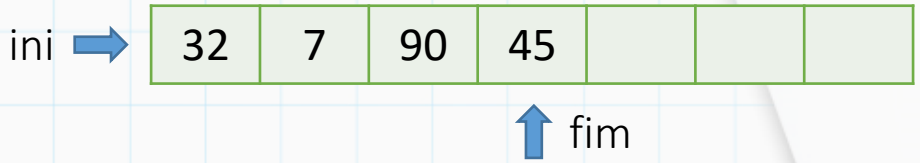


Filas

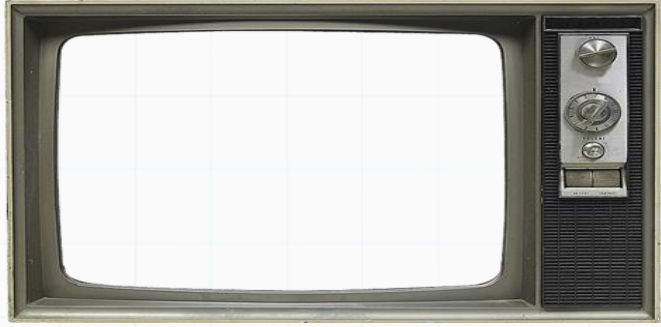


- TAD de Filas :
  - Dados:
    - Elementos (são os dados)
    - Ponteiro para o início e para o fim da fila
  - Operações:
    - entra(F, x): insere um elemento x no fim da fila (enfileira).
    - sai(F): retira o elemento no começo da fila (desenfileira).
    - primeiro(F): Checa o valor do primeiro da fila (sem retirar)
    - ultimo(F): Checa o valor do ultimo da fila (sem retirar)
    - Vazia(F): Checa se fila vazia.
    - Cheia(F): Checa se fila cheia.

outras operações

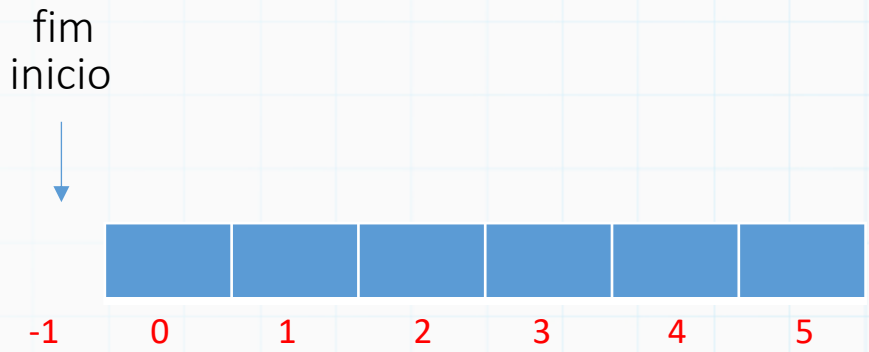


Filas

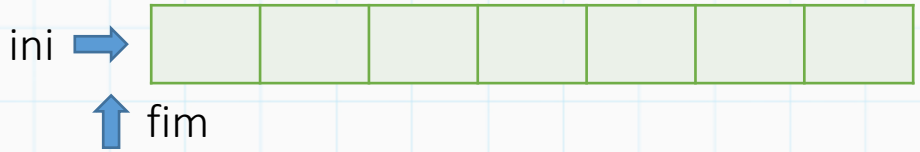


- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim

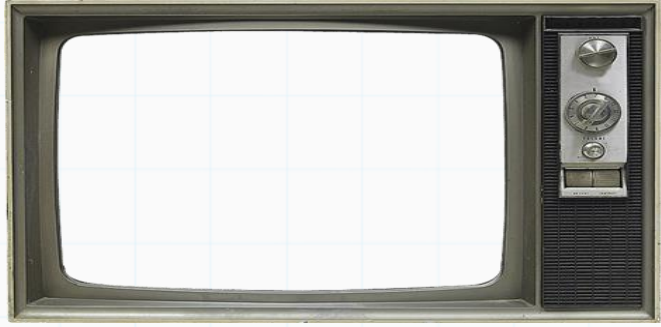
VETOR



REPRESENTAÇÃO GRÁFICA DE FILA

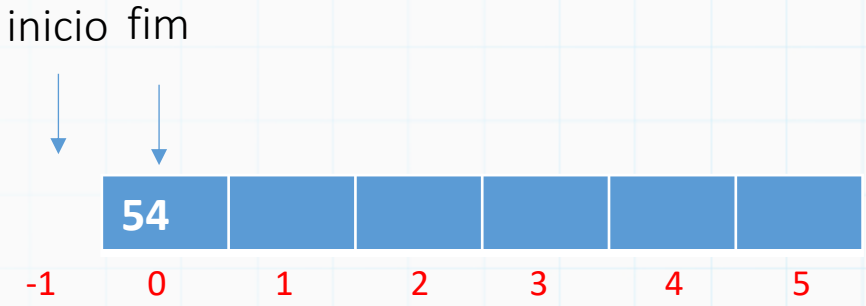


Filas

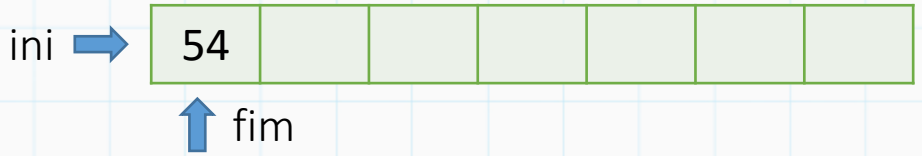


- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54-> entra(F,54)

VETOR

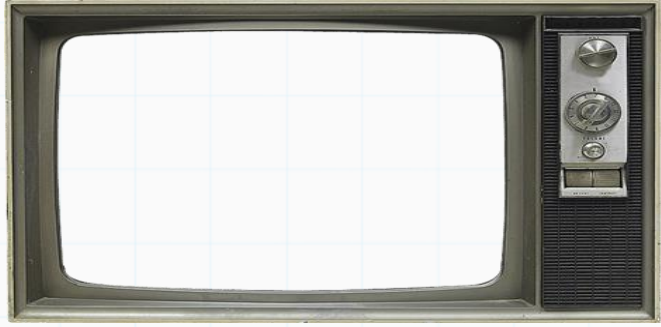


REPRESENTAÇÃO GRÁFICA DE FILA



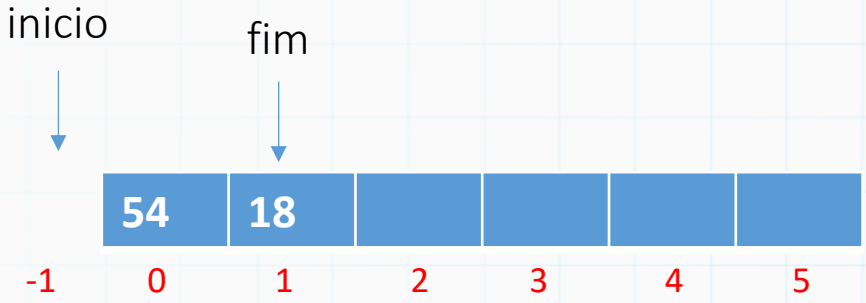
inicio aponta para antes do primeiro da fila

Filas

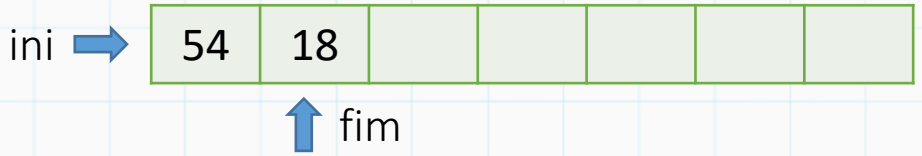


- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18-> entra(F,54), entra(F,18)

VETOR

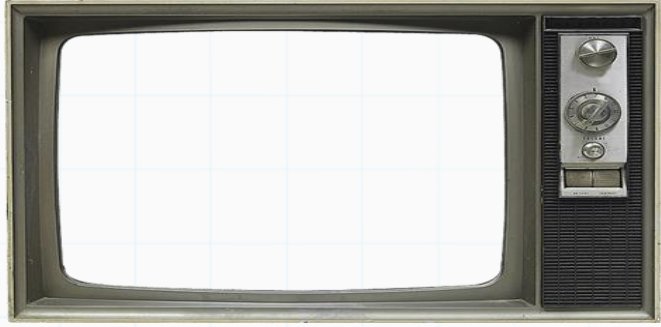


REPRESENTAÇÃO GRÁFICA DE FILA



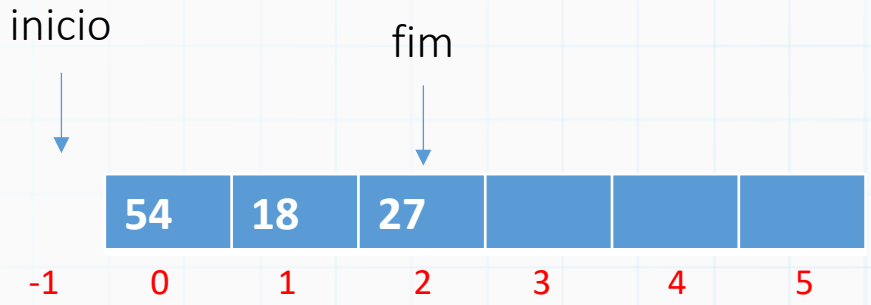
inicio aponta para antes do primeiro da fila

Filas

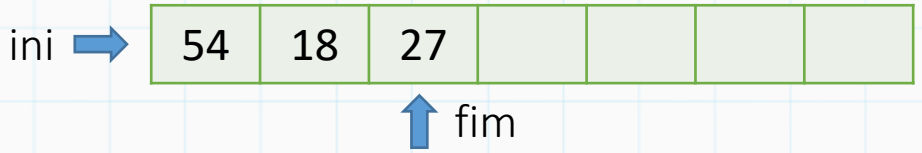


- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27-> entra(F,54), entra(F,18), entra(F,27)

VETOR

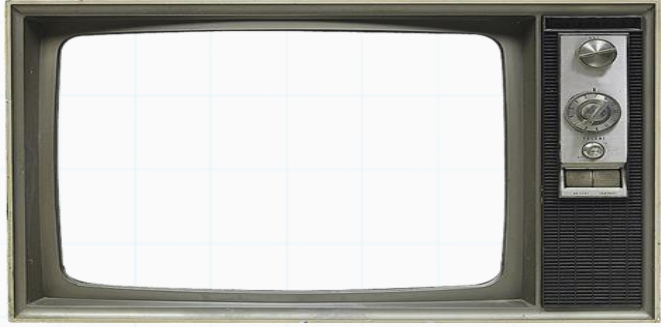


REPRESENTAÇÃO GRÁFICA DE FILA



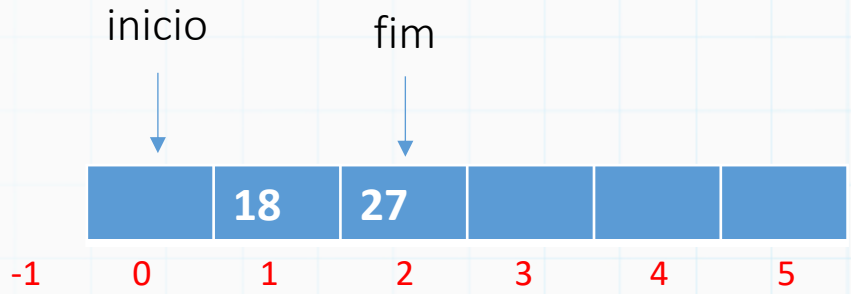
inicio aponta para antes do primeiro da fila

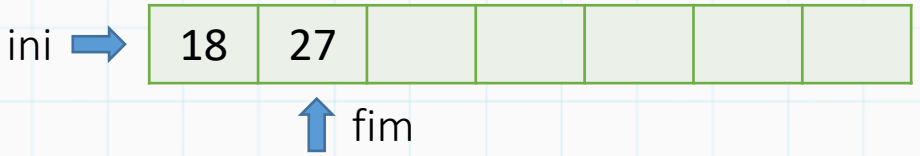
Filas



- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27-> entra(F,54), entra(F,18), entra(F,27)
  - Vamos agora retirar alguém da fila -> sai(F)

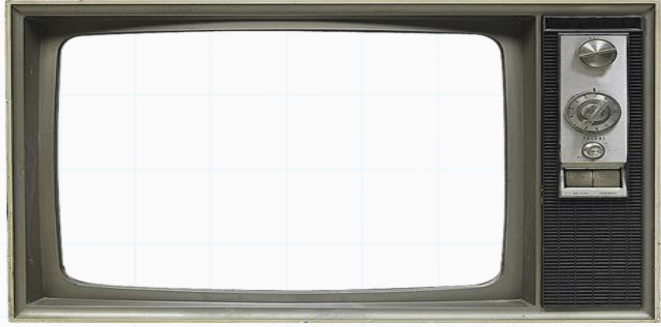
A fila está andando





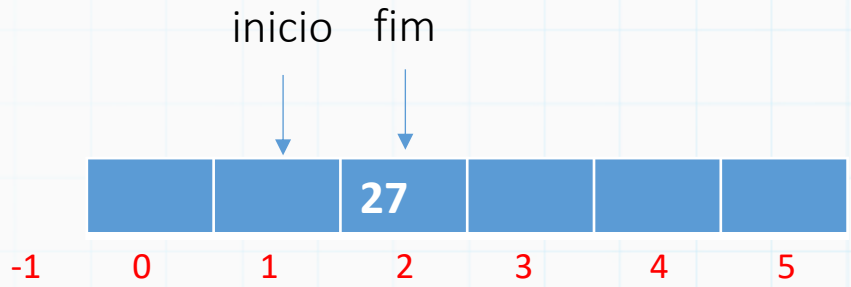
inicio aponta para antes do primeiro da fila

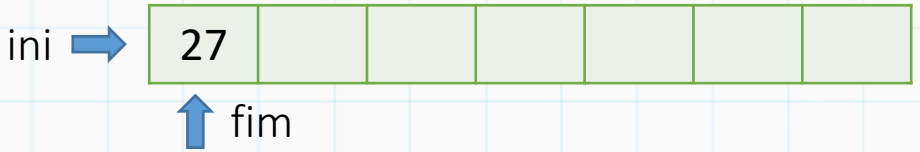
Filas



- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27-> entra(F,54), entra(F,18), entra(F,27)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F)

A fila está andando



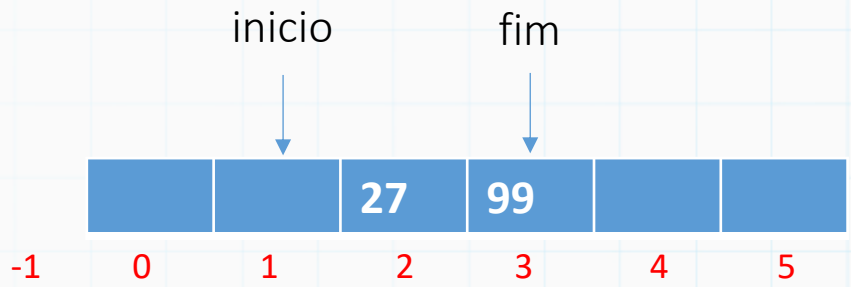


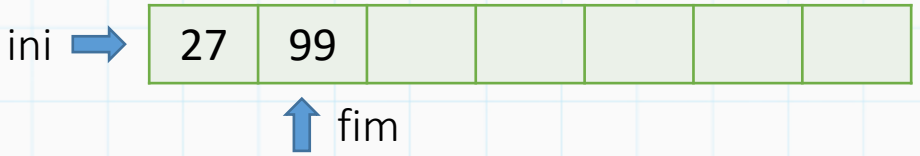
inicio aponta para antes do primeiro da fila

Filas

- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27,99-> entra(F,54), entra(F,18), entra(F,27), entra(F,99)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F)

A fila está andando





inicio aponta para antes do primeiro da fila

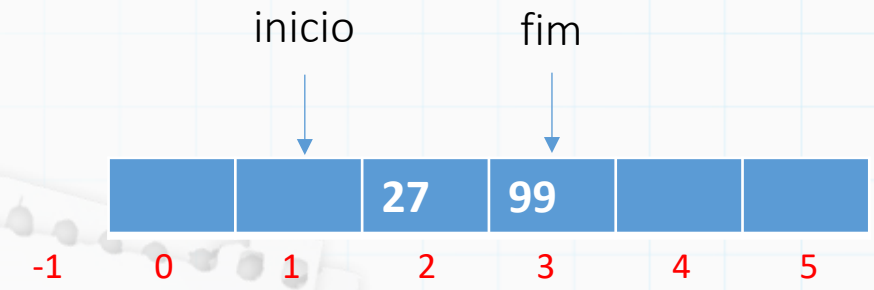
Filas

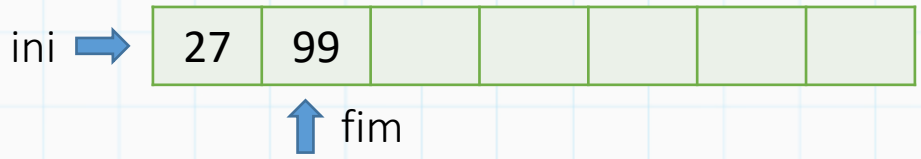
- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27,99-> entra(F,54), entra(F,18), entra(F,27), entra(F,99)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F)

```
struct FILA
{
int tam;
int ini;
int fim;
int *v;
};typedef struct FILA fila;
```

```
void cria_fila(fila* F, int tam)
{
     F->ini = -1;
     F->fim = -1;
     F->tam  = tam;
     F->v    = (int*) malloc( F->tam * sizeof(int));
}
```

```
int main()
{
    fila F;
    cria_fila(&F, 10);

     return 0;

}
```



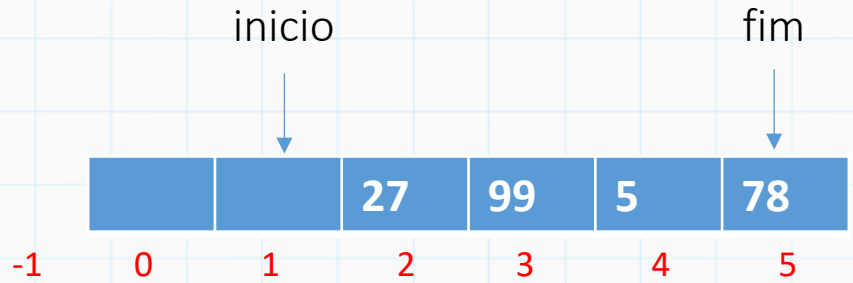


Filas

- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27,99-> entra(F,54), entra(F,18), entra(F,27), entra(F,99)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F)

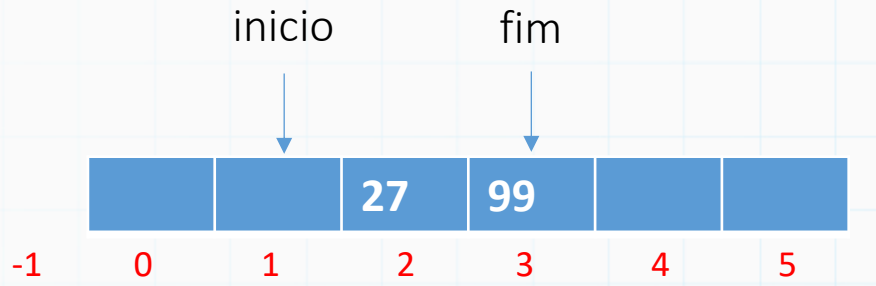
```
int vazia_fila(fila * F)
{
    if (F->ini == F->fim)
        return 1;
    else
        return 0;
}
```

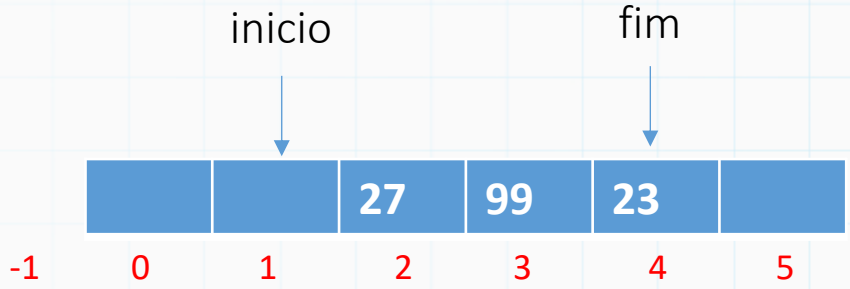
```
int cheia_fila(fila * F)
{
    if (F->fim == F->tam-1)
        return 1;
    else
        return 0;
}
```



Filas

- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27,99-> entra(F,54), entra(F,18), entra(F,27), entra(F,99)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F)

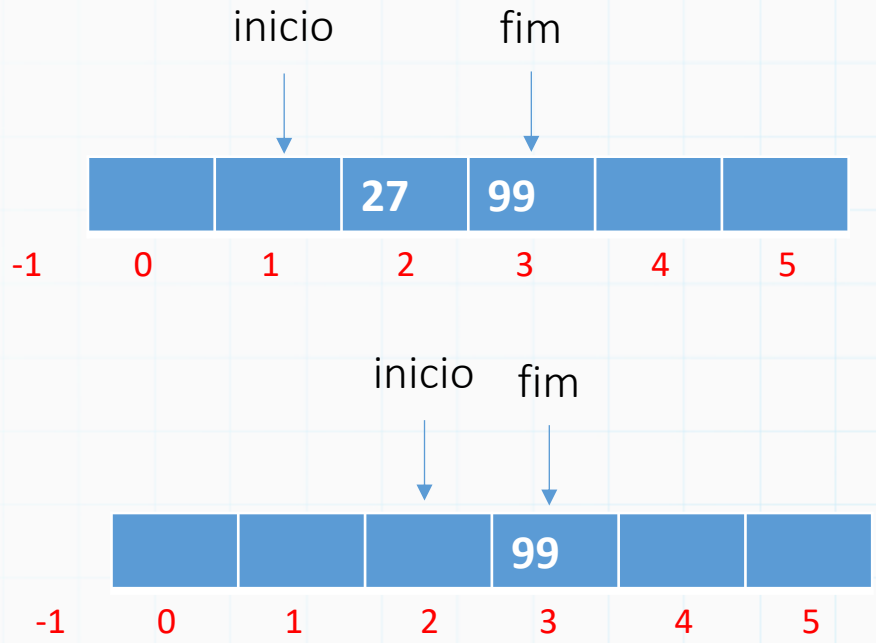





```
void entra_fila(fila * F, int el)
{
    if(cheia_fila(F) == 1)
        printf("fila cheia\n");
    else
    {
        F->fim++;
        F->v[F->fim] = el;
    }
}
```

Filas

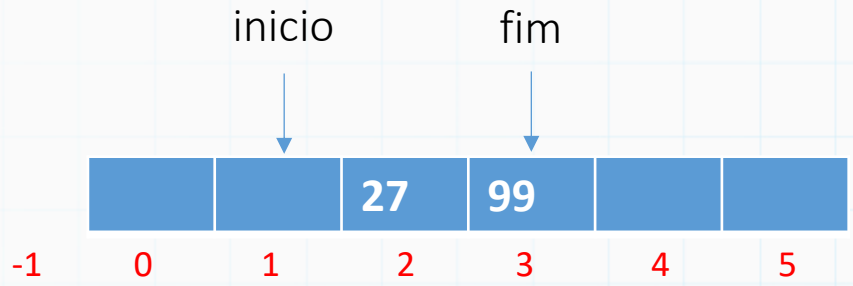
- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27,99-> entra(F,54), entra(F,18), entra(F,27), entra(F,99)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F)




```
int sai_fila(fila * F)
{
    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
    {
        F->ini++;
        return F->v[F->ini];
    }

    return -1;
}
```

Filas

- Filas em vetores:
  - Usaremos vetor com o tamanho máximo da fila (ex: 6) e dois índices, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 18, 27,99-> entra(F,54), entra(F,18), entra(F,27), entra(F,99)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F)

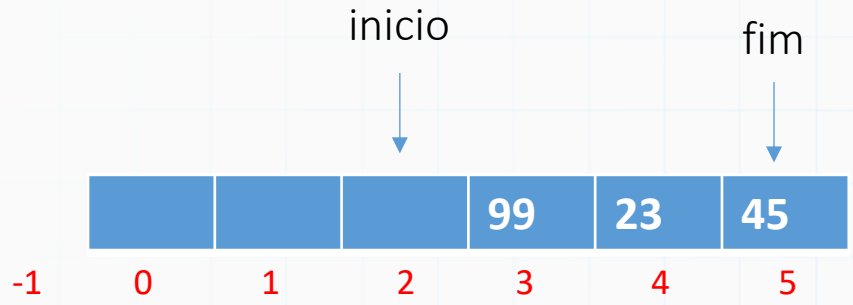



```
int primeiro_fila(fila * F)
{
    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
        return F->v[F->ini + 1];

    return -1;
}
```

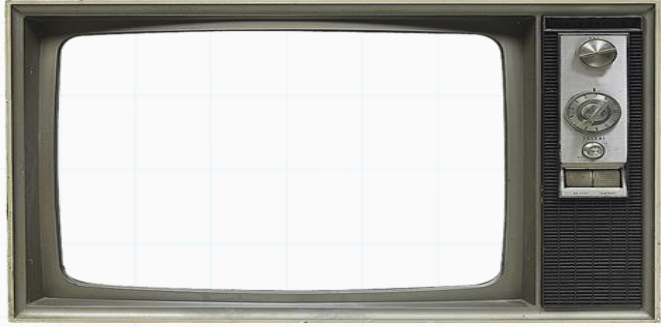
```
int ultimo_fila(fila * F)
{
    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
        return F->v[F->fim];

    return -1;
}
```

Filas

- Filas em vetores: Problemas



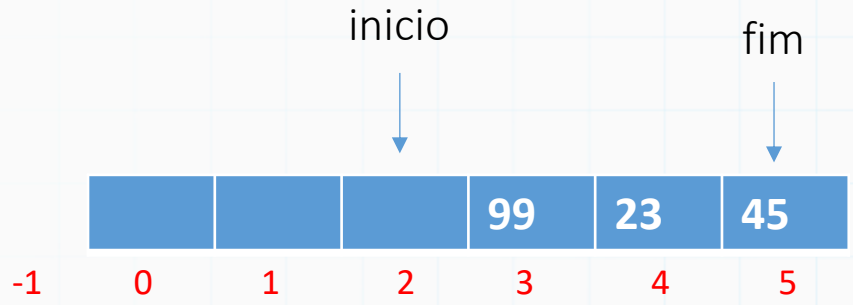
- Ao tentarmos inserir, a fila vai dizer que está cheia, mesmo tendo espaço antes do inicio.

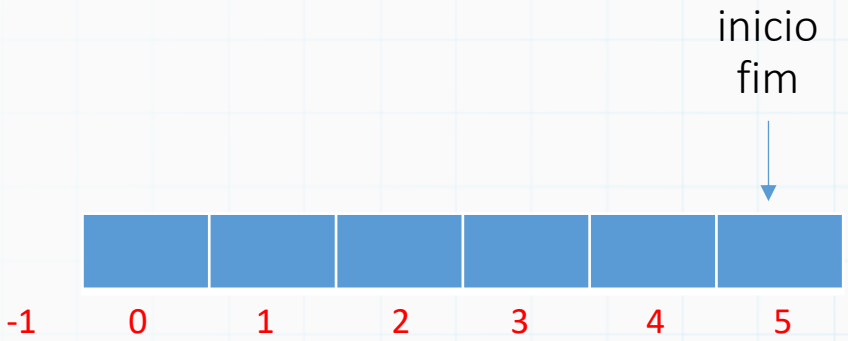


```
int cheia_fila(fila * F)
{
    if (F->fim == F->tam-1)
        return 1;
    else
        return 0;
}
```

Filas

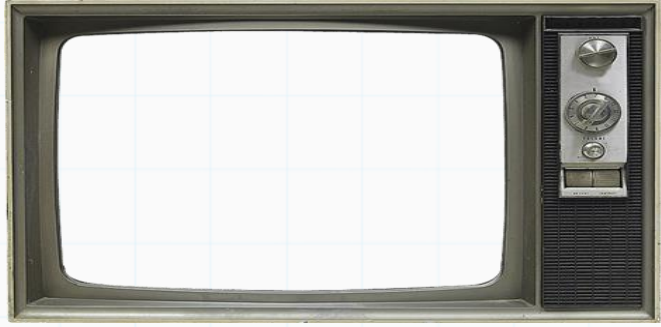
- Filas em vetores: Problemas





- Ao tentarmos inserir, a fila vai dizer que está cheia, mesmo tendo espaço antes do inicio.

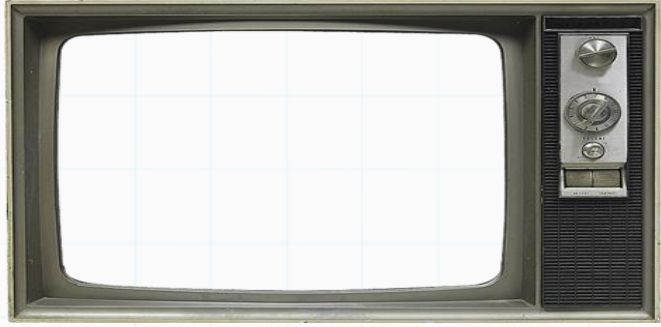
- Veja até que é possível estar cheia e vazia ao mesmo tempo



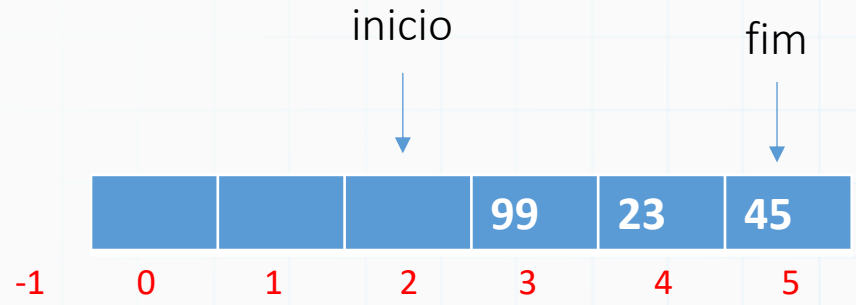
```
int cheia_fila(fila * F)
{
    if (F->fim == F->tam-1)
        return 1;
    else
        return 0;
}
```

```
int vazia_fila(fila * F)
{
    if (F->ini == F->fim)
        return 1;
    else
        return 0;
}
```

Filas

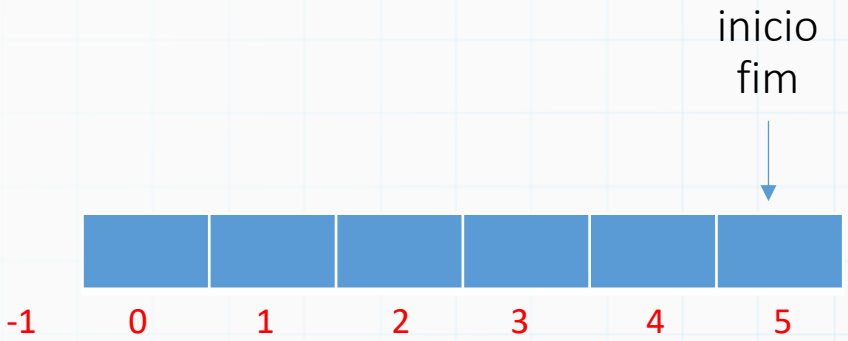


- Filas em vetores: Problemas



- Ao tentarmos inserir, a fila vai dizer que está cheia, mesmo tendo espaço antes do inicio.

```
int cheia_fila(fila * F)
{
    if (F->fim == F->tam-1)
        return 1;
    else
        return 0;
}
```

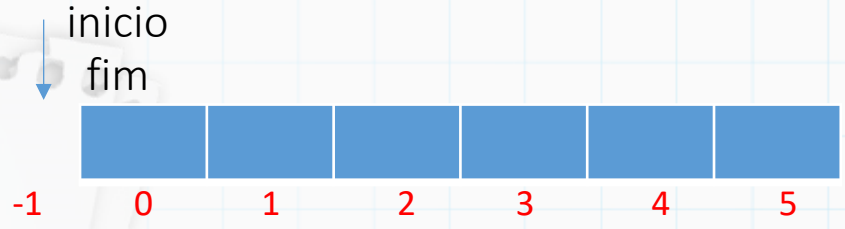


- Veja até que é possível estar cheia e vazia ao mesmo tempo

```
int vazia_fila(fila * F)
{
    if (F->ini == F->fim)
        return 1;
    else
        return 0;
}
```

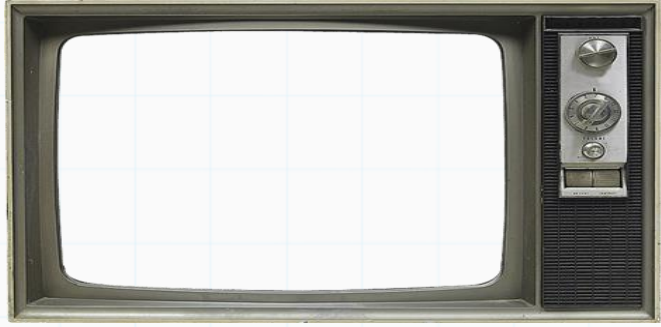
Soluções:

1) Podemos resetar os ponteiros de inicio e fim sempre que a fila estiver vazia

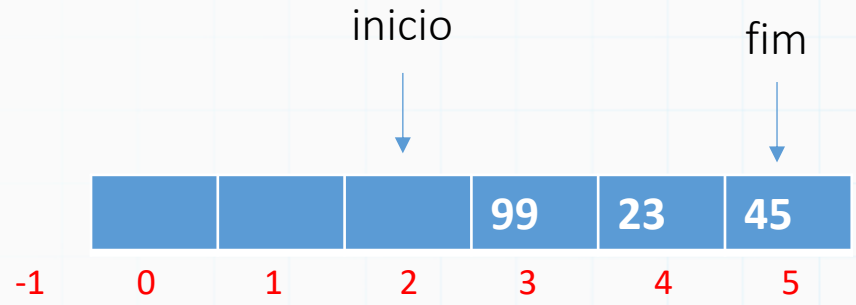


mas isso só resolve o problema se a fila ficar vazia

Filas

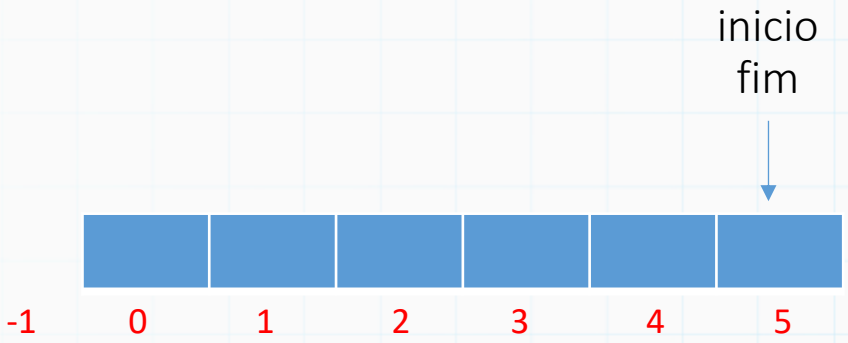


- Filas em vetores: Problemas



- Ao tentarmos inserir, a fila vai dizer que está cheia, mesmo tendo espaço antes do inicio.

```
int cheia_fila(fila * F)
{
    if (F->fim == F->tam-1)
        return 1;
    else
        return 0;
}
```



- Veja até que é possível estar cheia e vazia ao mesmo tempo

```
int vazia_fila(fila * F)
{
    if (F->ini == F->fim)
        return 1;
    else
        return 0;
}
```

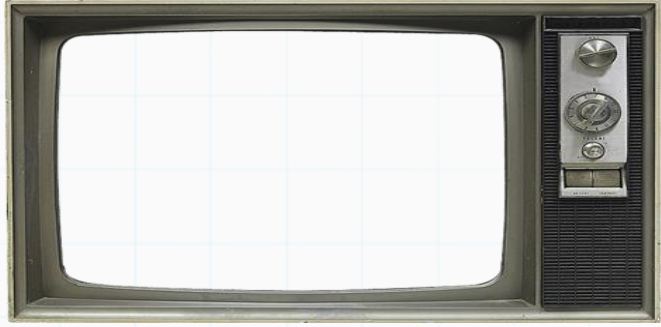
Soluções:

1) Podemos resetar os ponteiros de inicio e fim sempre que a fila estiver vazia

2) Filas circulares

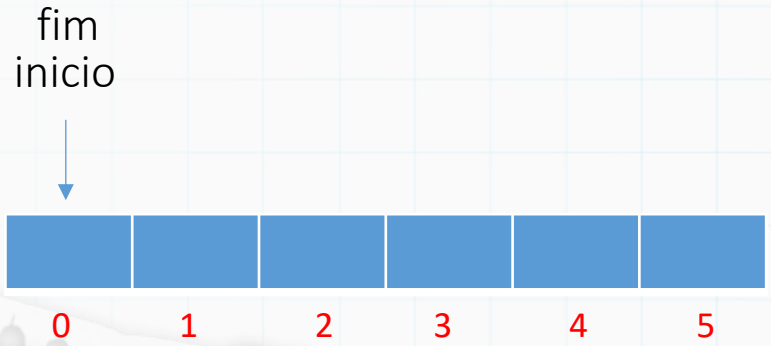
Filas

- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor

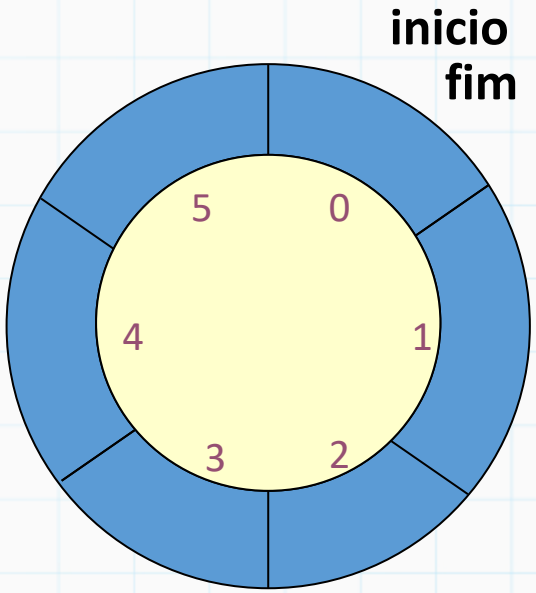


```
struct FILA
{
int tam;
int ini;
int fim;
int *v;
};typedef struct FILA fila;
```

```
void cria_fila(fila* F, int tam)
{
     F->ini = 0;
     F->fim = 0;
     F->tam  = tam;
     F->v    = (int*) malloc( F->tam * sizeof(int));
}
```

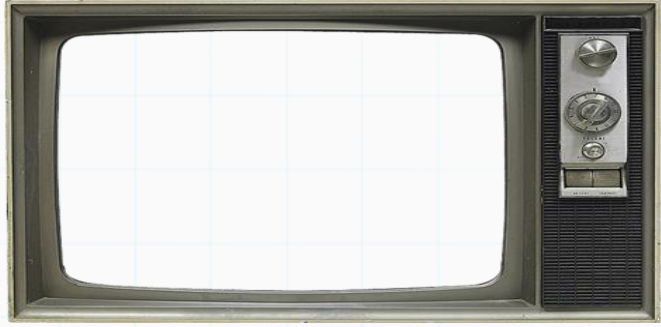


VETOR

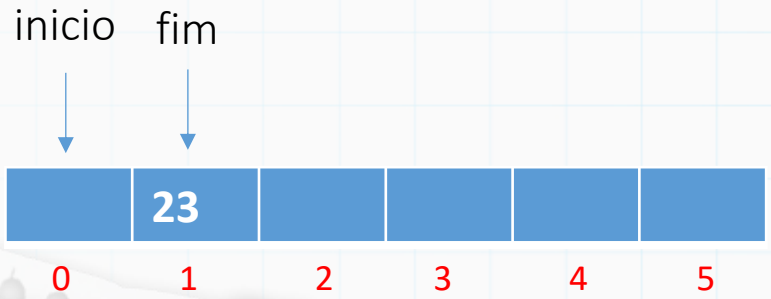


REPRESENTAÇÃO GRÁFICA DE FILA

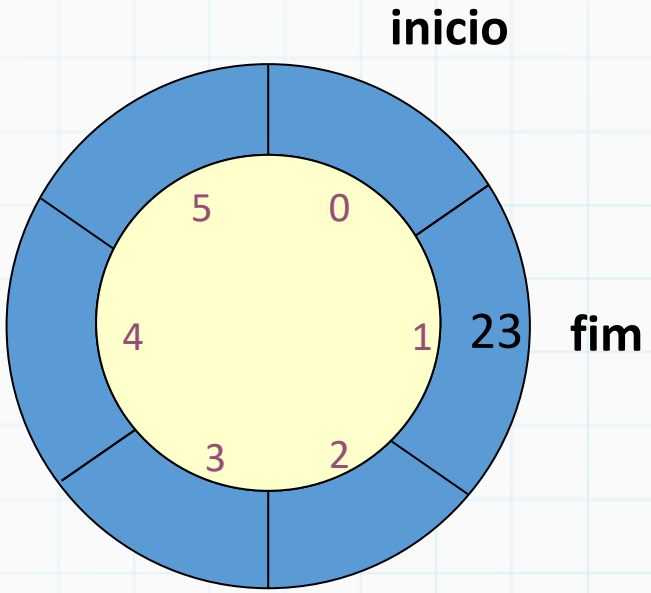
Filas



- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:

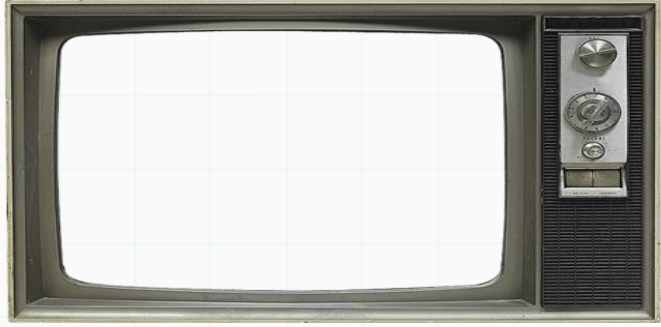


VETOR

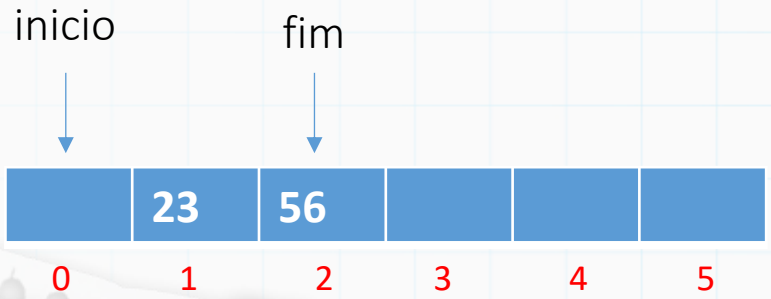


REPRESENTAÇÃO GRÁFICA DE FILA

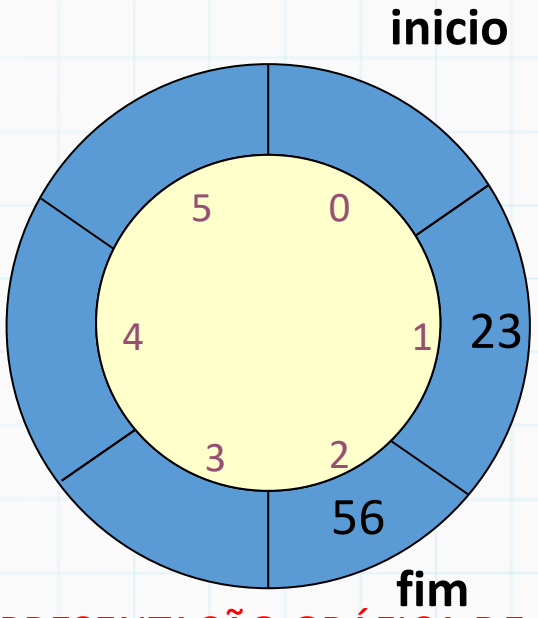
Filas



- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:

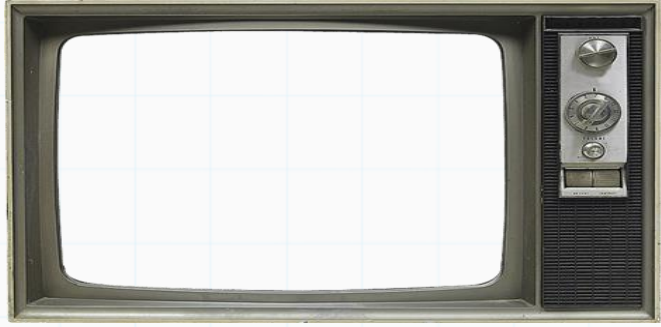


VETOR

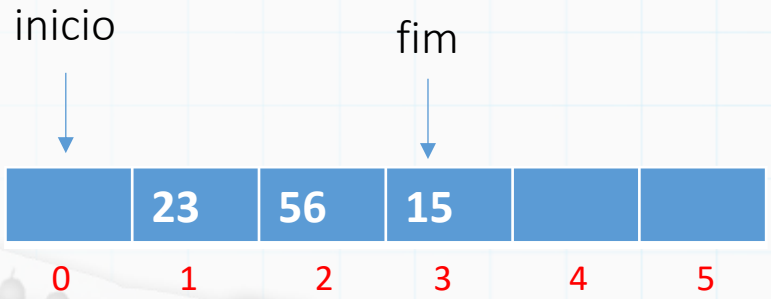


REPRESENTAÇÃO GRÁFICA DE FILA

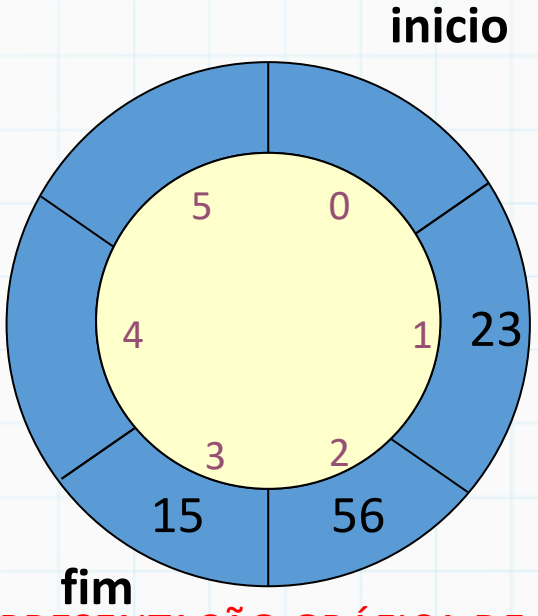
Filas



- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:

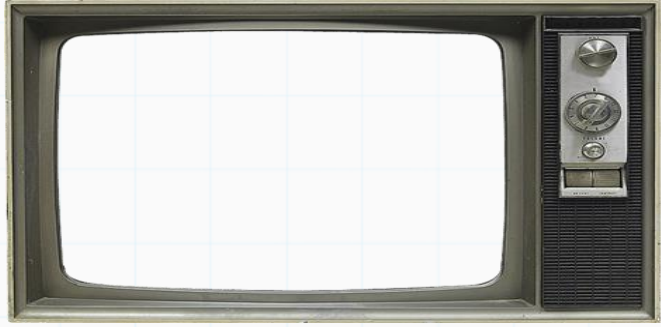


VETOR

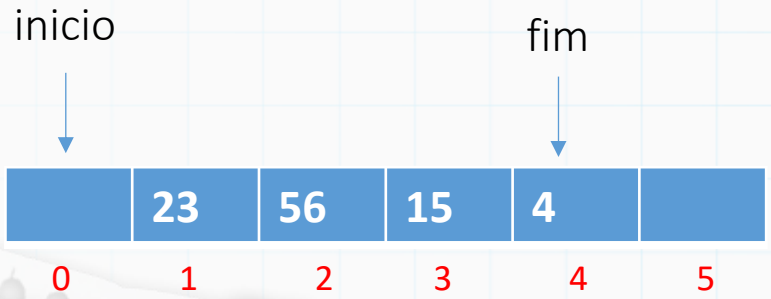


REPRESENTAÇÃO GRÁFICA DE FILA

Filas



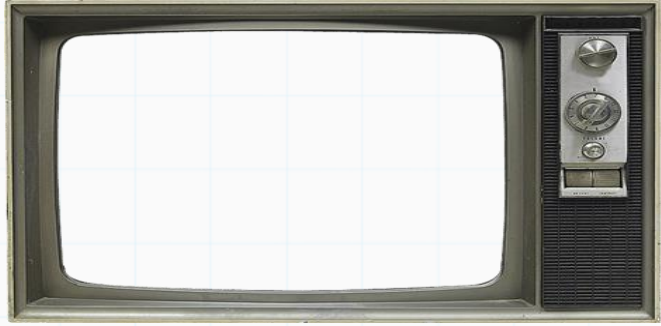
- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:



VETOR

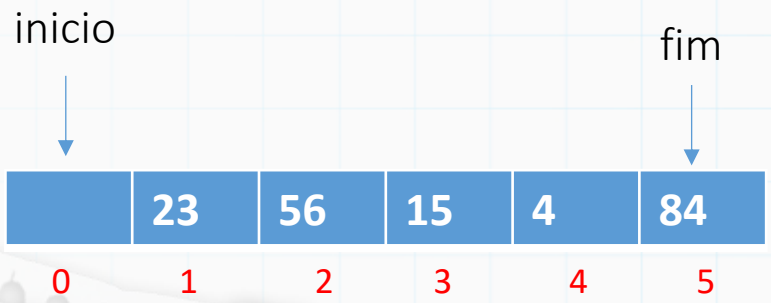
REPRESENTAÇÃO GRÁFICA DE FILA

Filas

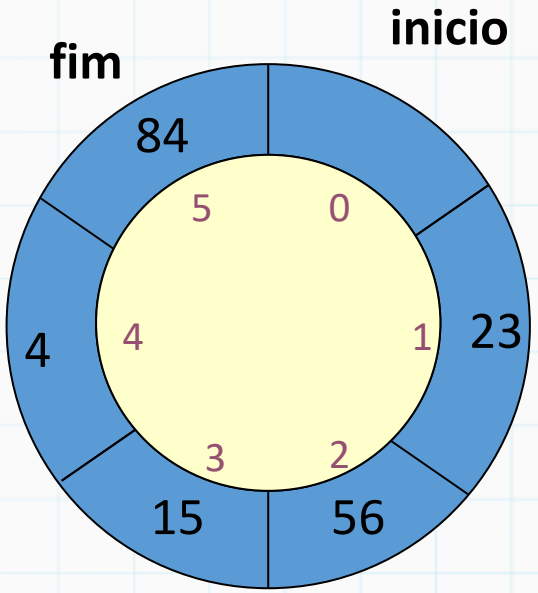


- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:

Vamos retirar um elemento agora

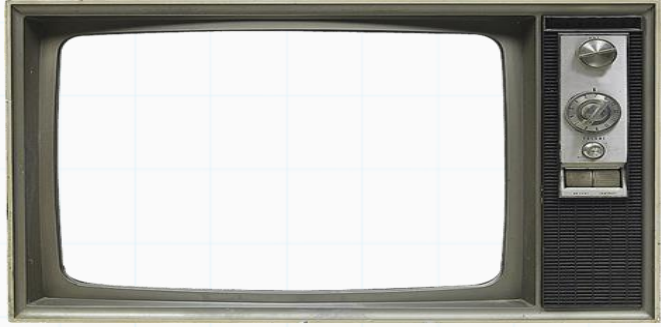


VETOR



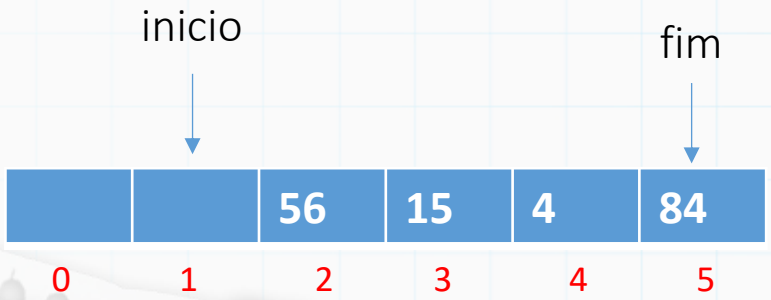
REPRESENTAÇÃO GRÁFICA DE FILA

Filas

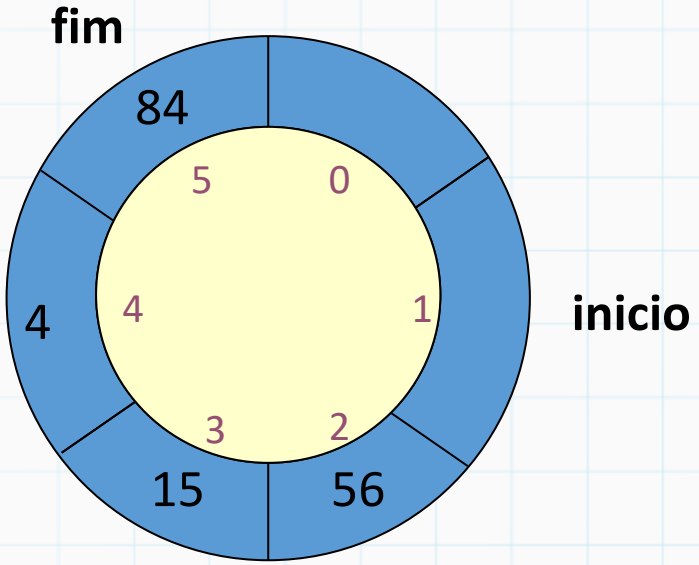


- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:
- Retirar elementos:

Vamos inserir de novo

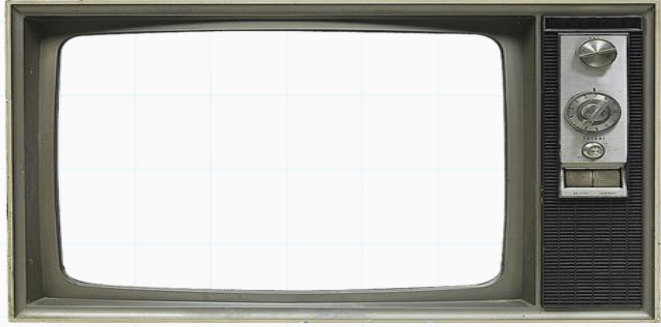


VETOR



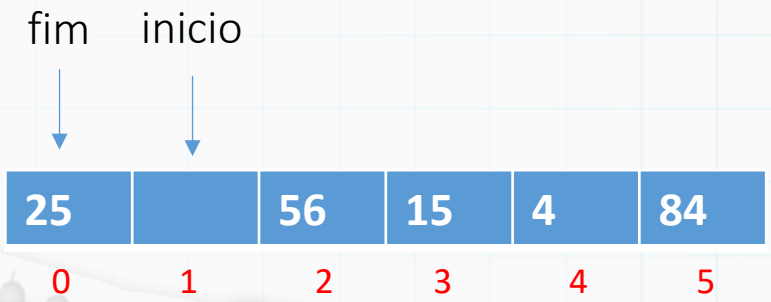
REPRESENTAÇÃO GRÁFICA DE FILA

Filas

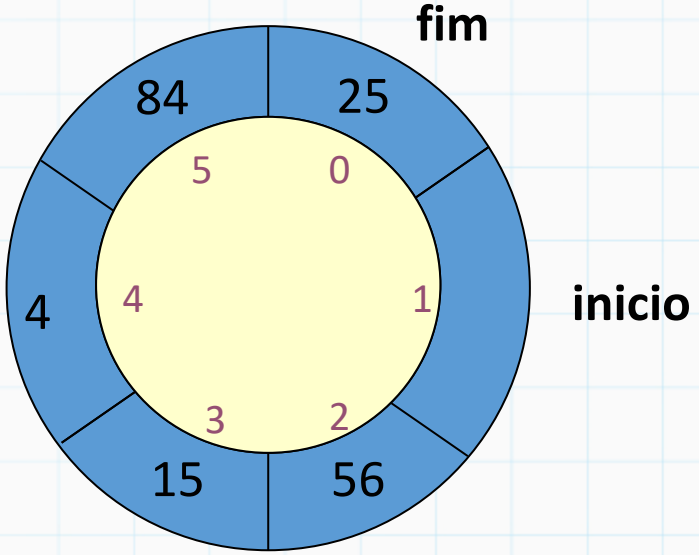


- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:
- Retirar elementos:
- Inserir o elemento 25, mas e agora ?

Vamos inserir de novo

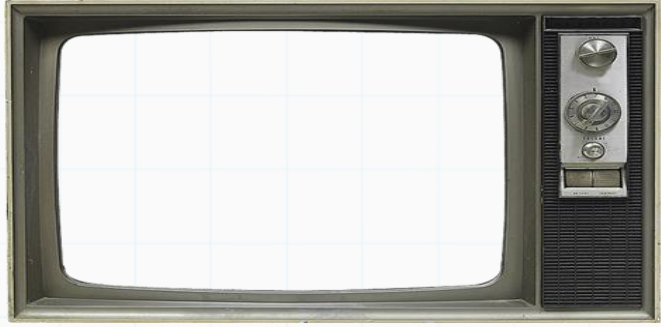


VETOR



REPRESENTAÇÃO GRÁFICA DE FILA

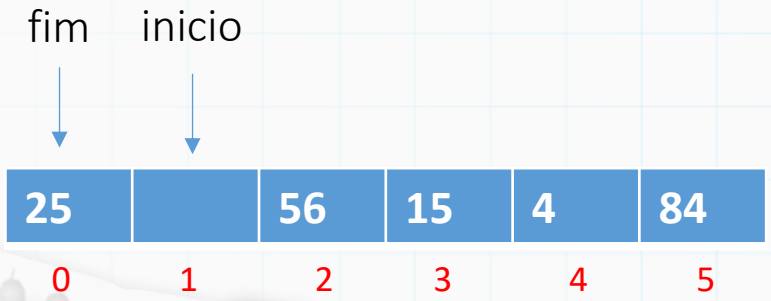
Filas



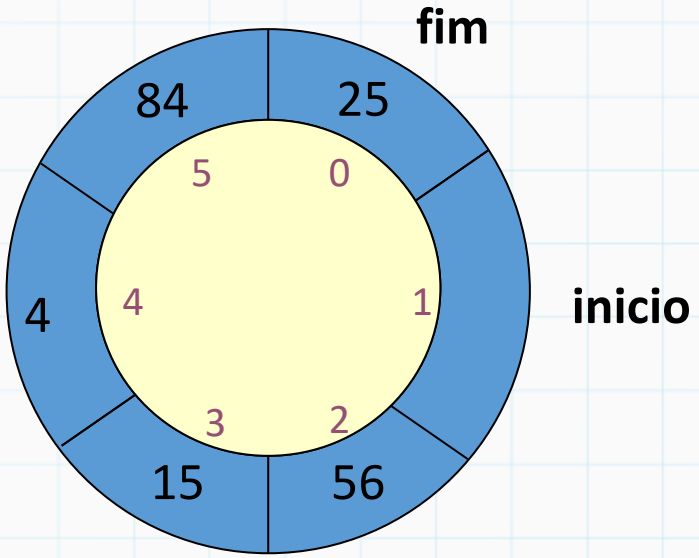
- Filas em vetores: Filas circulares -> simula um comportamento circular em um vetor
- Vamos inserir elementos agora:
- Retirar elementos:
- Inserir o elemento 25, mas e agora ?
- Inserir o elemento 52 ?

Não podemos inserir mais pois senão o fim alcança o inicio, consideramos a fila cheia

A fila mesmo cheia fica com uma posição vazia para marcar o início



VETOR



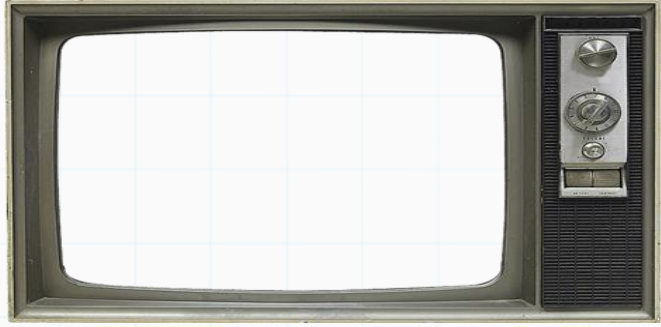
REPRESENTAÇÃO GRÁFICA DE FILA

Filas

- Filas em vetores: Filas circulares

```
int vazia_fila(fila * F)
{
    if (F->ini == F->fim)
        return 1;
    else
        return 0;
}
```

```
int cheia_fila(fila * F)
{
    if ((F->fim+1)%F->tam == F->ini)
        return 1;
    else
        return 0;
}
```



Veja que para TAM=6

0%6 = 0

1%6 = 1

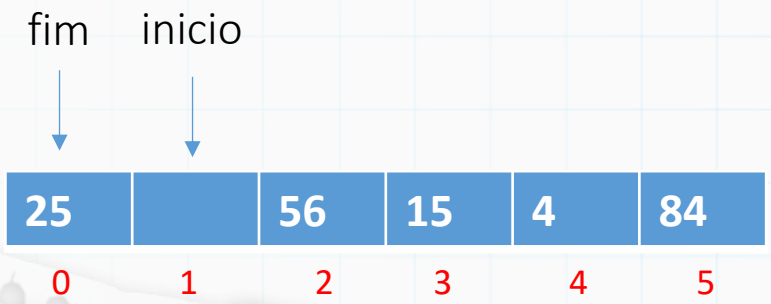
2%6 =2

3%6 = 3

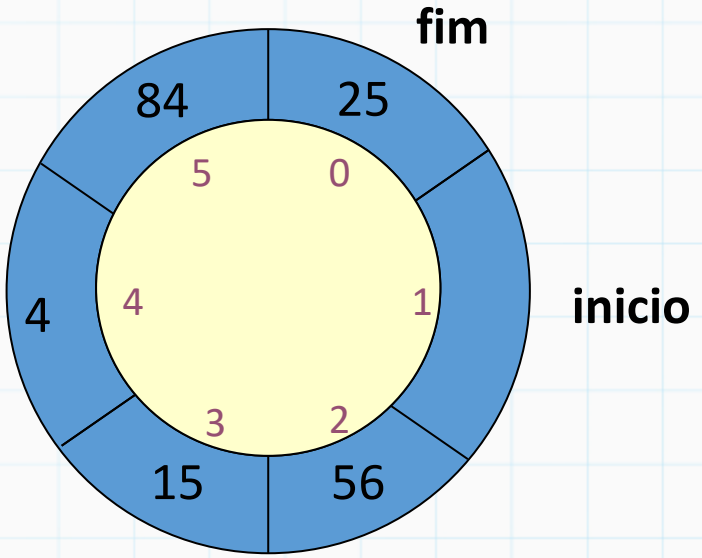
4%6 = 4

5%6 = 5

6%6 = 0



VETOR

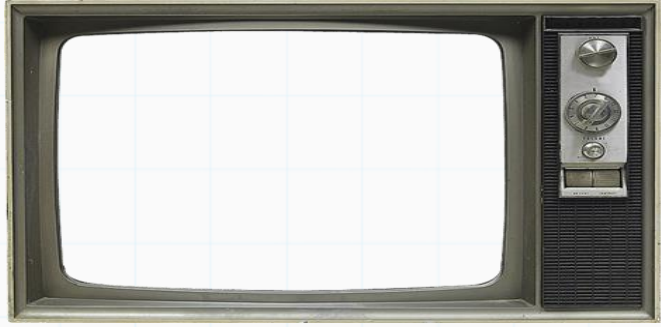


REPRESENTAÇÃO GRÁFICA DE FILA

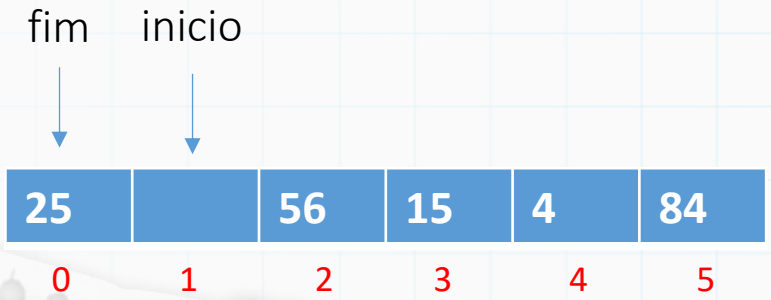
Filas

- Filas em vetores: Filas circulares

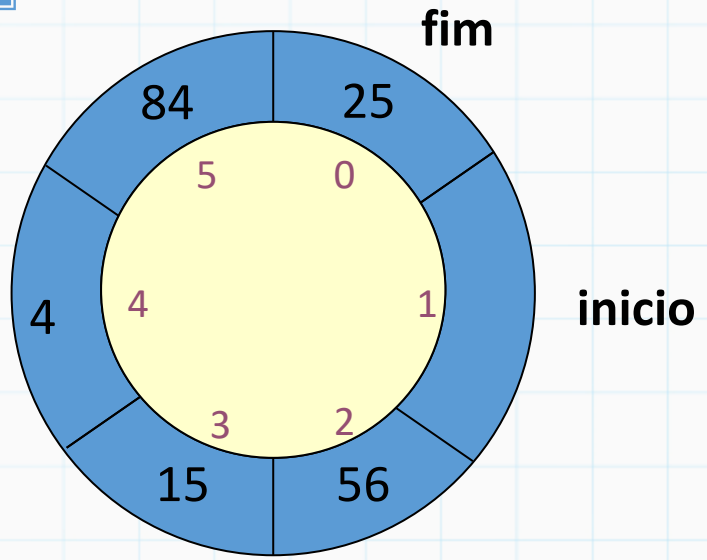
```
void entra_fila(fila * F, int el)
{
    if(cheia_fila(F) == 1)
        printf("fila cheia\n");
    else
    {
        F->fim = ((F->fim+1)%F->tam);
        F->v[F->fim] = el;
    }
}
```



Veja que para TAM=6
0%6 = 0
1%6 = 1
2%6 =2
3%6 = 3
4%6 = 4
5%6 = 5
6%6 = 0



VETOR

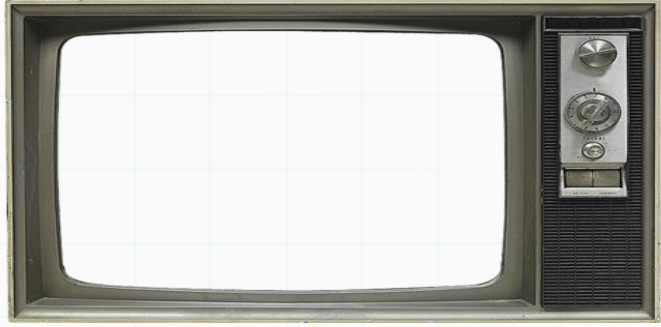


REPRESENTAÇÃO GRÁFICA DE FILA

Filas

- Filas em vetores: Filas circulares

```
int sai_fila(fila * F)
{
    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
    {
        F->ini = (F->ini+1)%F->tam;
        return F->v[F->ini];
    }

    return -1;
}
```



Veja que para TAM=6

0%6 = 0

1%6 = 1

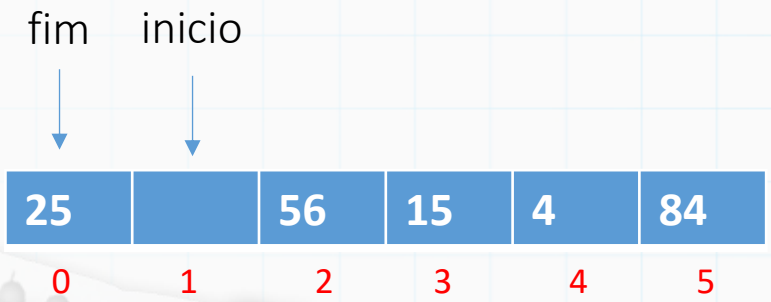
2%6 =2

3%6 = 3

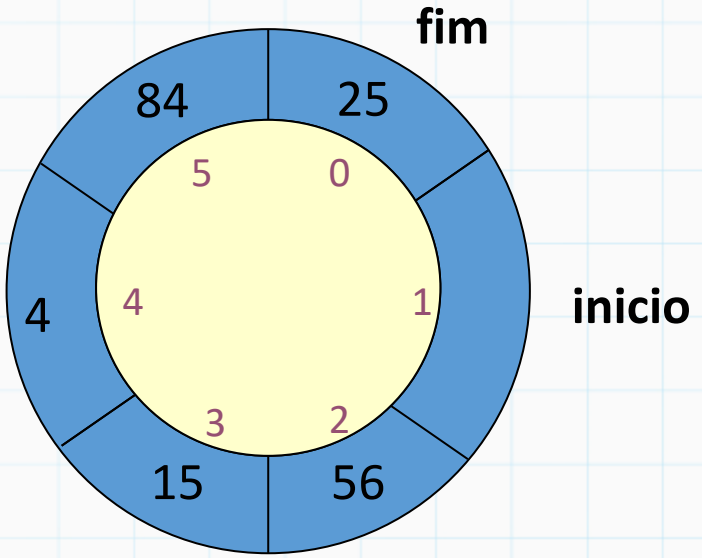
4%6 = 4

5%6 = 5

6%6 = 0



VETOR



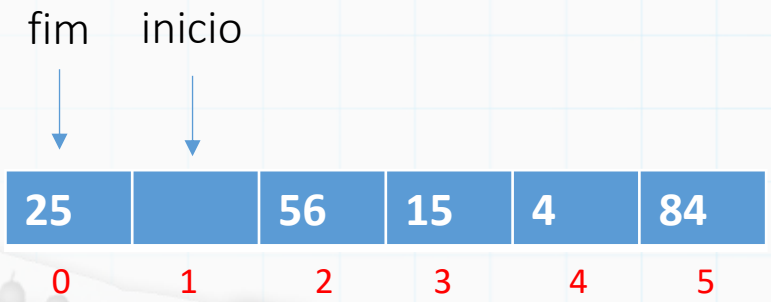
REPRESENTAÇÃO GRÁFICA DE FILA

Filas

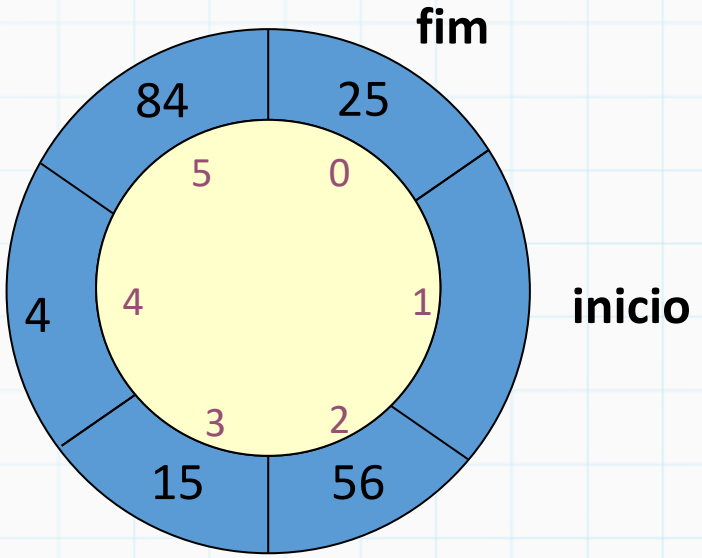
- Filas em vetores: Filas circulares

```
int primeiro_fila(fila * F)
{
    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
        return F->v[(F->ini+1)%F->tam];

    return -1;
}
```

```
int ultimo_fila(fila * F)
{
    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
        return F->v[((F->fim+1)%F->tam)];

    return -1;
}
```



VETOR



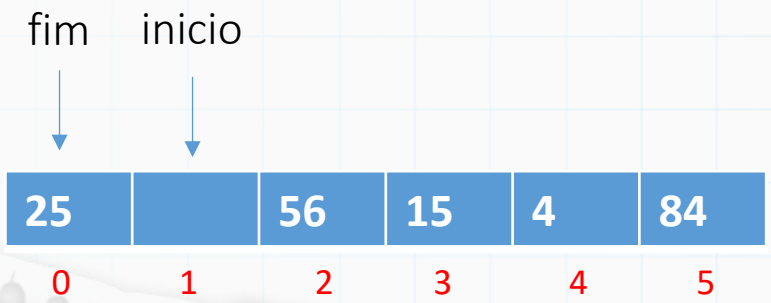
REPRESENTAÇÃO GRÁFICA DE FILA

Filas

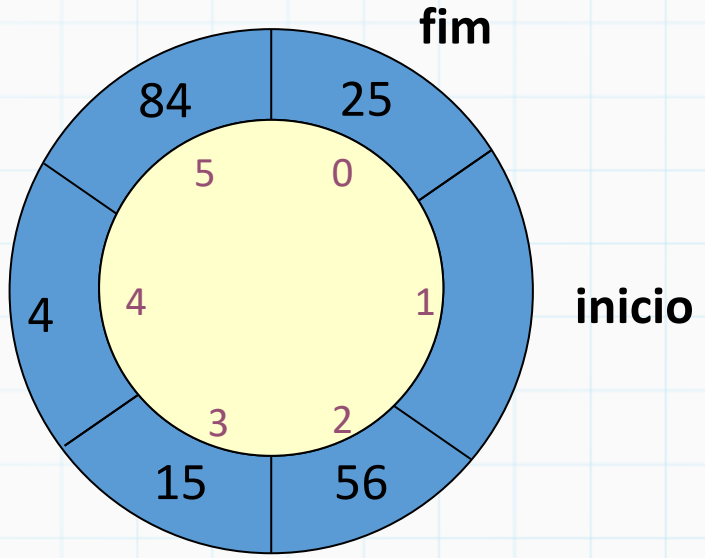
- Exercício 1) Dada uma fila circular, escreva uma função para imprimir seus elementos.

```
struct FILA
{
int tam;
int ini;
int fim;
int *v;
};typedef struct FILA fila;
```

```
void cria_fila(fila* F, int tam)
{
     F->ini = 0;
     F->fim = 0;
     F->tam  = tam;
     F->v    = (int*) malloc( F->tam * sizeof(int));
}
```



VETOR



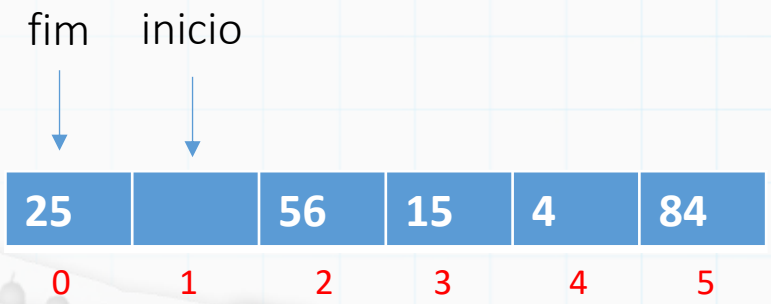
REPRESENTAÇÃO GRÁFICA DE FILA

Cuidado com o primeiro e o ultimo elemento da fila

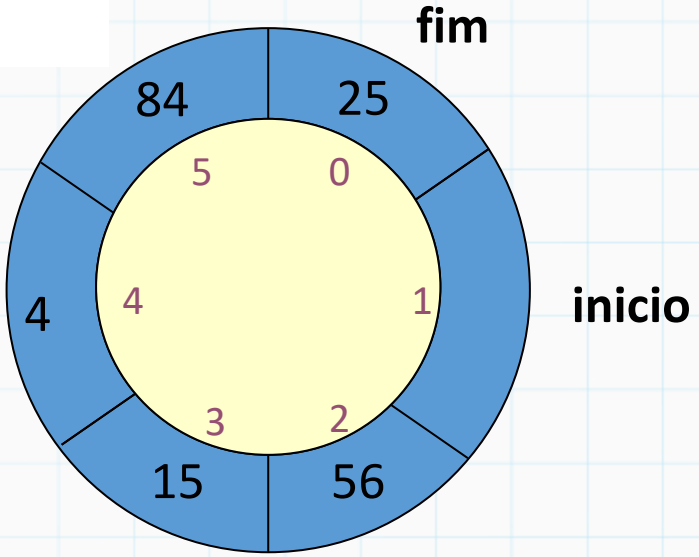
Filas

- Exercício 1) Dada uma fila circular, escreva uma função para imprimir seus elementos.

```c
void imprime_fila(fila * F)
{
    int i = (F->ini+1)%F->tam;

    do
    {
        printf("| %2d |",F->v[i]);
        i = (i + 1)%F->tam;
    }
    while (i != (F->fim+1)%F->tam);
    printf("\n");
}
```



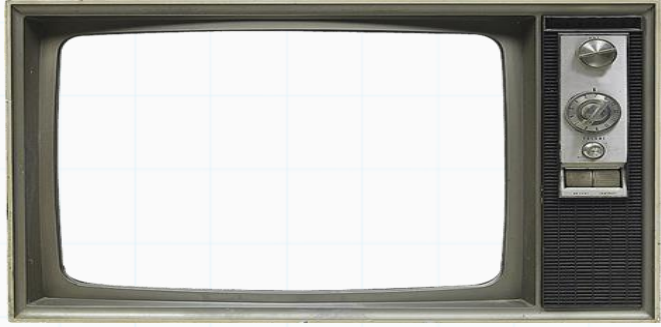
VETOR



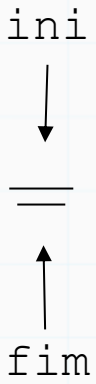
REPRESENTAÇÃO GRÁFICA DE FILA

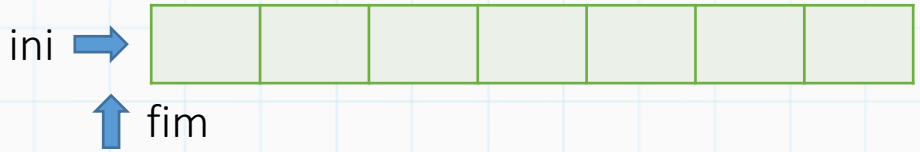
Cuidado com o primeiro e o ultimo elemento da fila

Filas

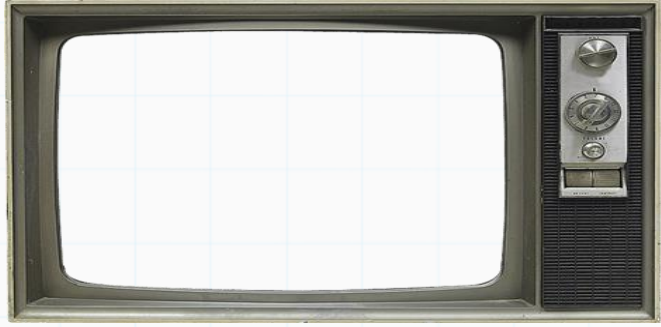


- Filas em listas encadeadas:
  - Usaremos dois ponteiros, uma para apontar para o inicio e outro para o fim

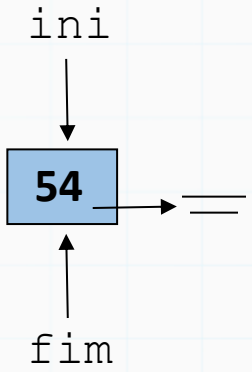


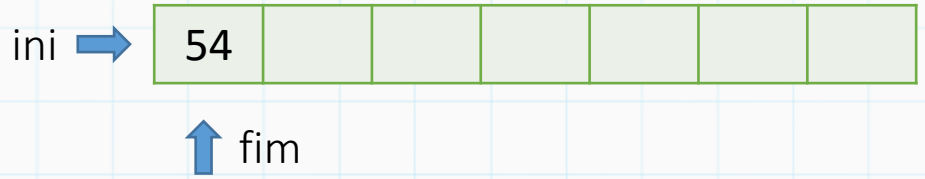


Filas

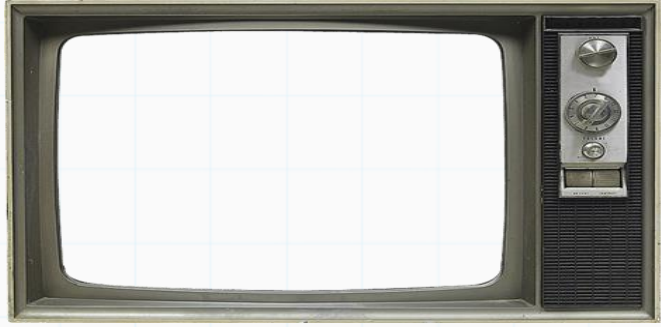


- Filas em listas encadeadas:
  - Usaremos dois ponteiros, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54 -> entra(F,54)

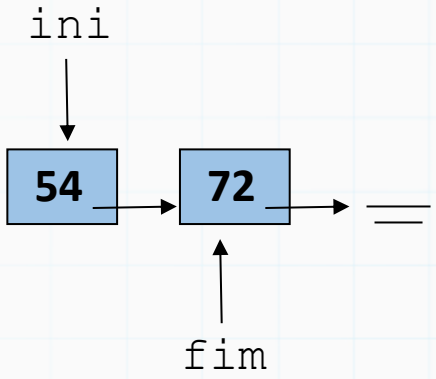


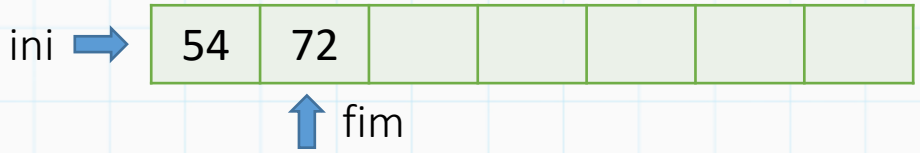


Filas

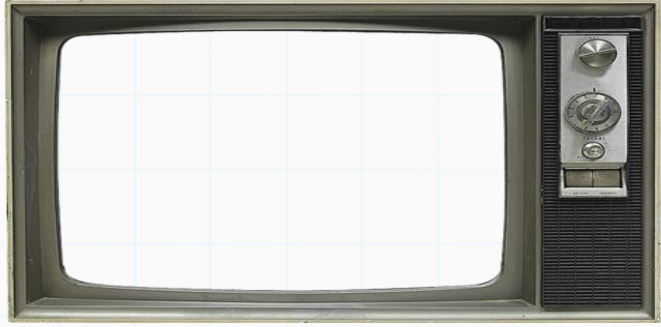


- Filas em listas encadeadas:
  - Usaremos dois ponteiros, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 72 -> entra(F,54), entra(F,72)

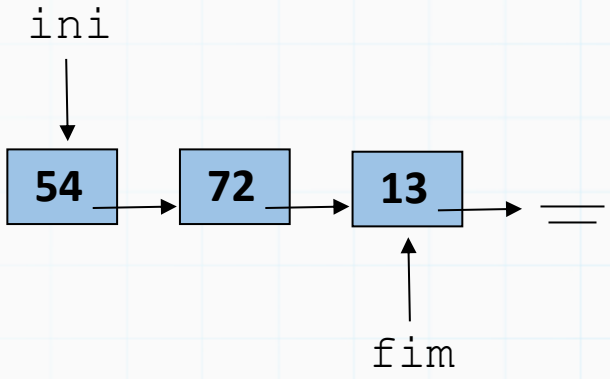


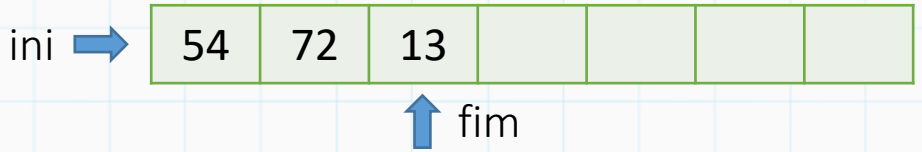


Filas



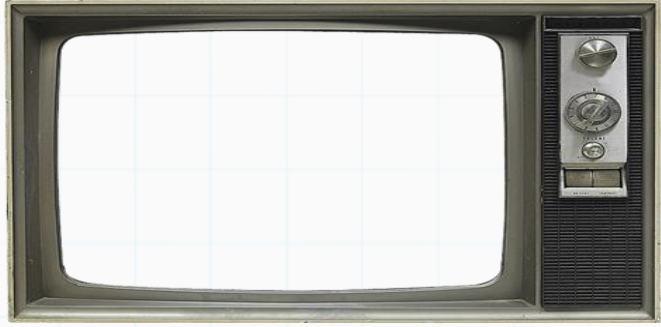
- Filas em listas encadeadas:
  - Usaremos dois ponteiros, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 72, 13 -> entra(F,54), entra(F,72), entra(F,13)



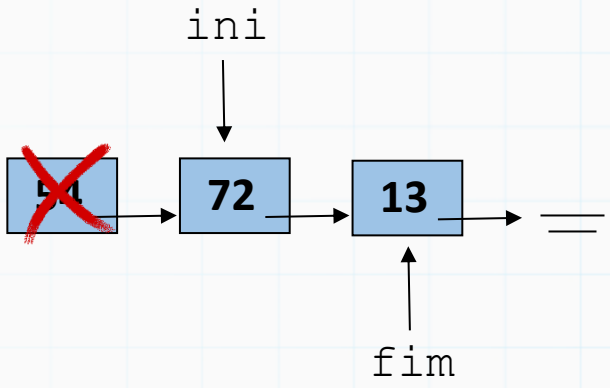


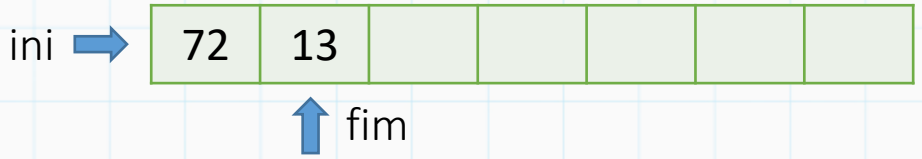
Vamos tirar da fila agora

Filas



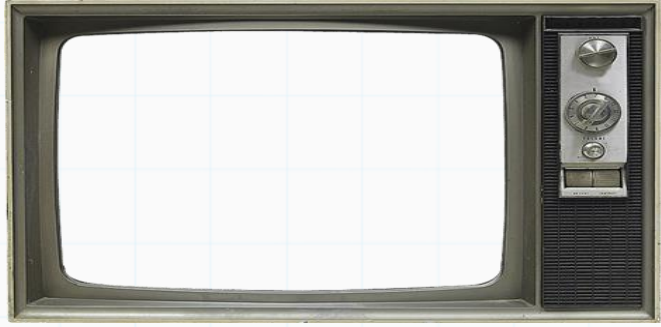
- Filas em listas encadeadas:
  - Usaremos dois ponteiros, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 72, 13 -> entra(F,54), entra(F,72), entra(F,13)
  - Vamos agora retirar alguém da fila -> sai(F),



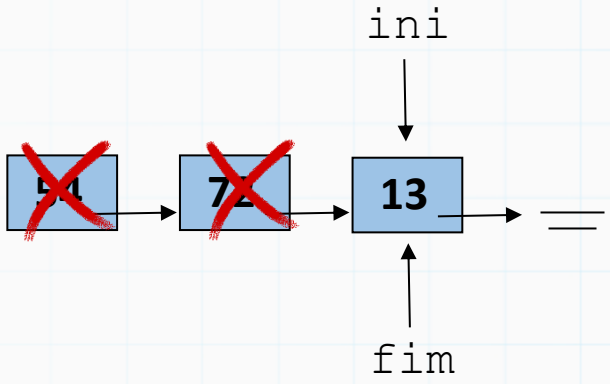


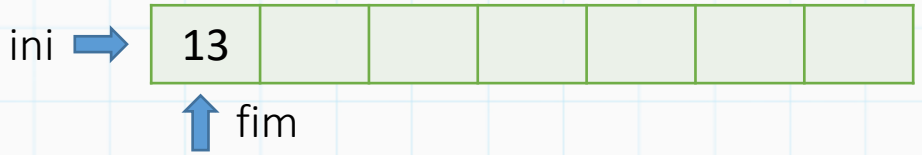
Vamos tirar mais um

Filas



- Filas em listas encadeadas:
  - Usaremos dois ponteiros, uma para apontar para o inicio e outro para o fim
  - Vamos enfileirar elemento 54, 72, 13 -> entra(F,54), entra(F,72), entra(F,13)
  - Vamos agora retirar alguém da fila -> sai(F), sai(F),





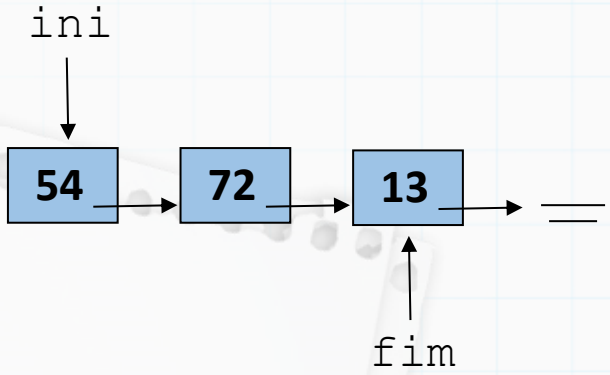
Filas

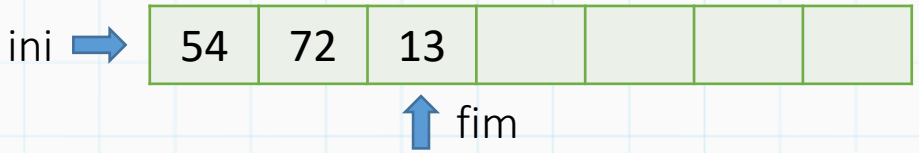
- Filas em listas encadeadas:

```
struct NO
{
        int info;
        struct NO *prox;
};
typedef struct NO no;
```

```
struct FILA
{
        struct NO *ini;
        struct NO *fim;
};
typedef struct FILA fila;
```

Utilizamos uma estrutura separada para fila pois temos que guardar/alterar/retornar dois ponteiros que identificam a fila (fica mais fácil assim)





Filas

- Filas em listas encadeadas:

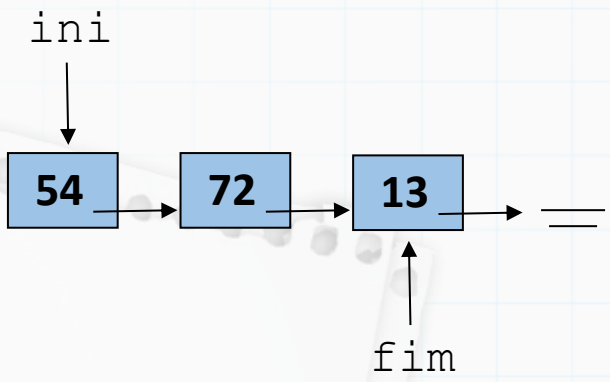
```
struct NO
{
        int info;
        struct NO *prox;
};
typedef struct NO no;
```

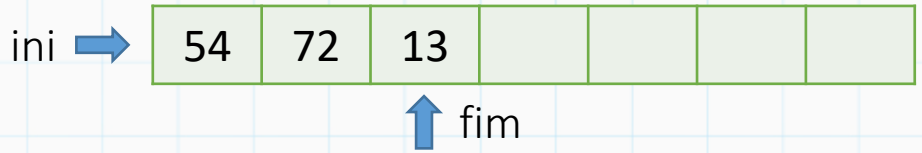
```
struct FILA
{
        struct NO *ini;
        struct NO *fim;
};
typedef struct FILA fila;
```

Utilizamos uma estrutura separada para fila pois temos que guardar/alterar/retornar dois ponteiros que identificam a fila (fica mais fácil assim)

```
no * aloca_no(void)
{
    no *aux;
    aux       = (no *) malloc (sizeof(no));
    aux->prox = NULL;

    return aux;
}
```

```
fila * aloca_fila(void)
{
    fila *aux;
    aux       = (fila *) malloc (sizeof(fila));
    aux->ini  = NULL;
    aux->fim  = NULL;

    return aux;
}
```



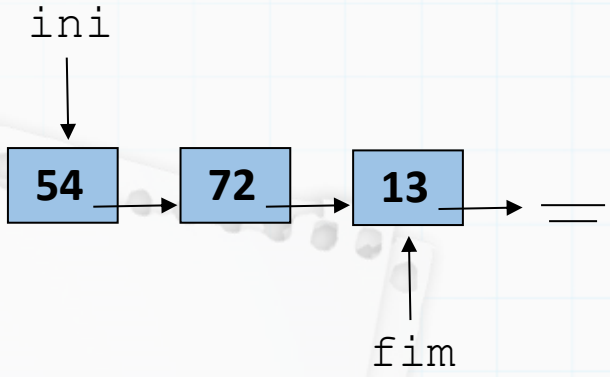


Filas

- Filas em listas encadeadas:

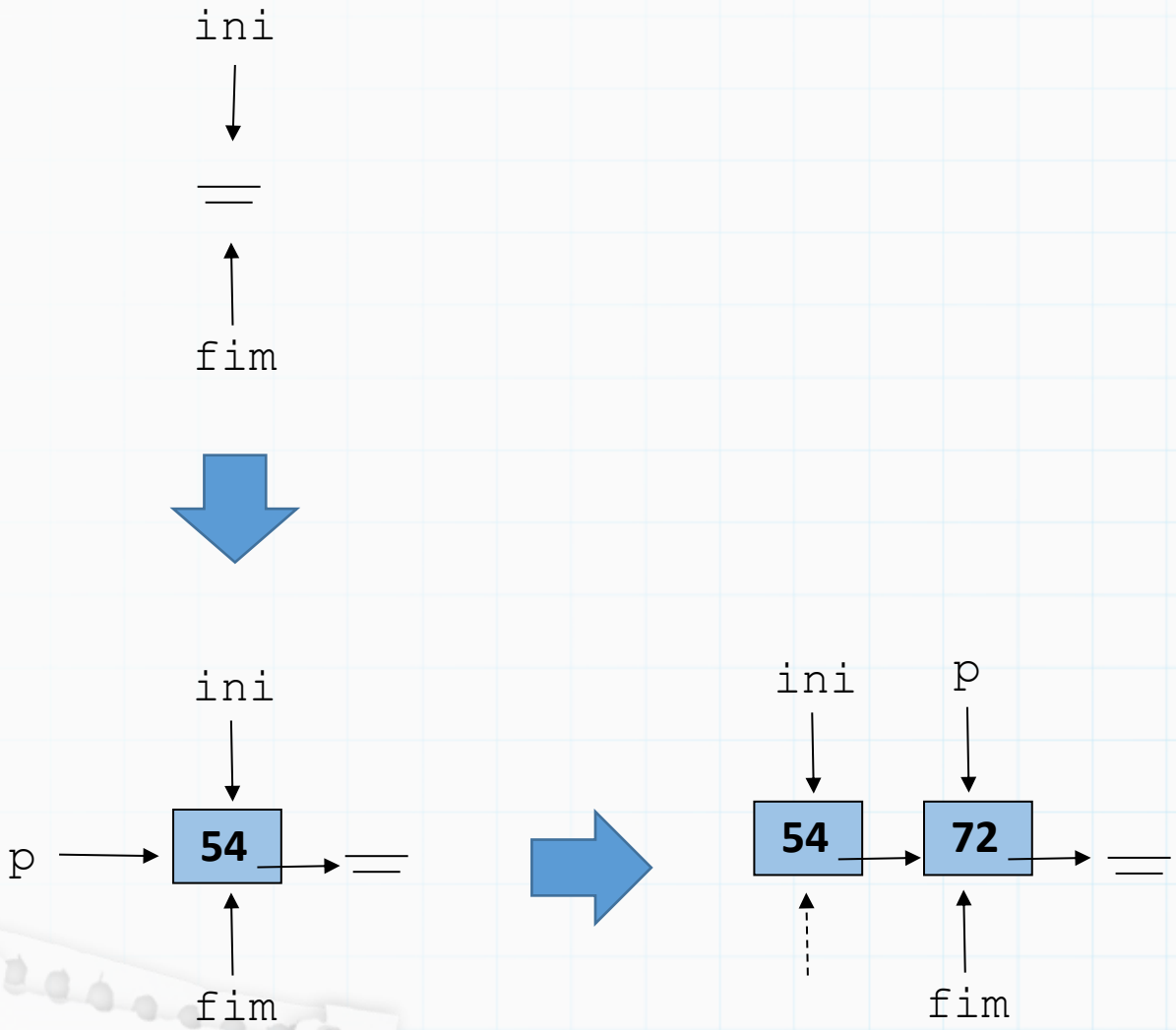
```
int vazia_fila(fila * F)
{
    if (F->ini==NULL)
        return 1;
    else
        return 0;
}
```

```
void imprime_fila(fila * F)
{
    no* p = F->ini;

    while (p != NULL)
    {
        printf("| %2d |",p->info);
        p = p->prox;
    }
    printf("\n");
}
```



Filas

- Filas em listas encadeadas:




```
void entra_fila(fila *F, int el)
{
    no *p;
    p        = aloca_no();
    p->info  = el;

    if (vazia_fila(F) == 1)
    {
        F->ini = p;
        F->fim = p;
    }
    else
    {
        F->fim->prox = p;
        F->fim       = p;
    }
}
```

Filas

- Exercício 2: Dada um fila em lista encadeada, escreva uma função que remova da fila e outra para apagar a fila toda, sabendo que:

```
struct NO
{
        int info;
        struct NO *prox;
};
typedef struct NO no;
```

```
struct FILA
{
        struct NO *ini;
        struct NO *fim;
};
typedef struct FILA fila;
```

```
int main()
{
    fila *F = aloca_fila();
    entra_fila(F, 2);
    entra_fila(F, 6);
    entra_fila(F, 7);
    entra_fila(F, 8);
    entra_fila(F, 20);
    printf("saiu = %d\n",sai_fila(F));
    imprime_fila(F);
    F = exclui_fila (F);
```

```
void entra_fila(fila *F, int el)
{
    no *p;
    p       = aloca_no();
    p->info  = el;

    if (vazia_fila(F) == 1)
    {
        F->ini = p;
        F->fim = p;
    }
    else
    {
        F->fim->prox = p;
        F->fim       = p;
    }
}
```

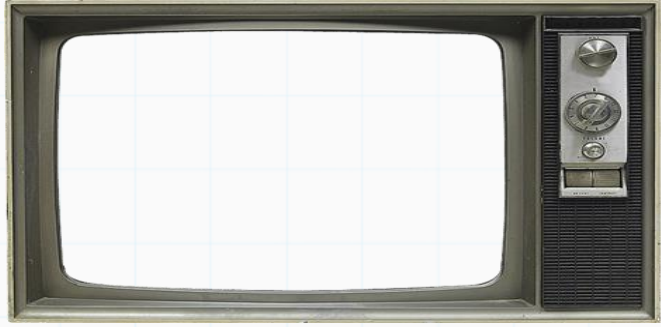
Filas

- Exercício 2: Dada um fila em lista encadeada, escreva uma função que remova da fila e outra para apagar a fila toda, sabendo que:

```c
int sai_fila(fila *F)
{
    int el;
    no  *p;

    if (vazia_fila(F) == 1)
    {
        printf("fila vazia\n");
        return -1;
    }
    else
    {
        el      = F->ini->info;
        p       = F->ini;
        F->ini = F->ini->prox;
        free(p);
        return el;
    }
}
```

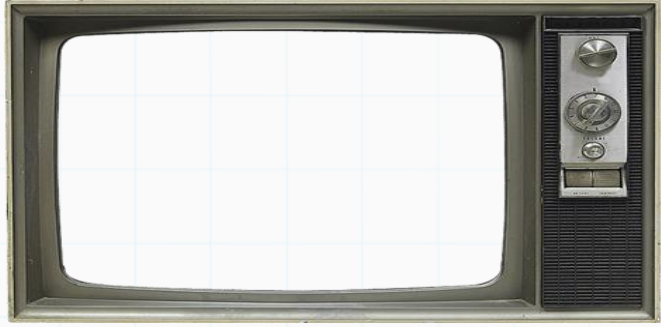
```c
fila * exclui_fila (fila* F)
{
    no* p = F->ini;
    while (p != NULL)
    {
        no* temp = p->prox;
        free(p);
        p        = temp;
    }
    free(F);

    return NULL;
}
```

Filas

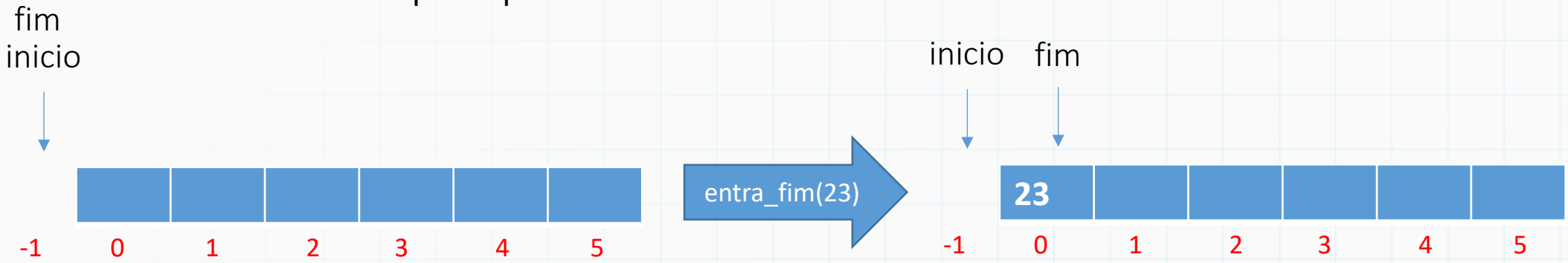


- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.

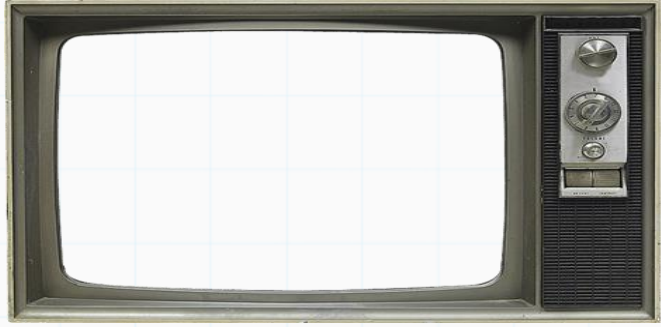
Filas



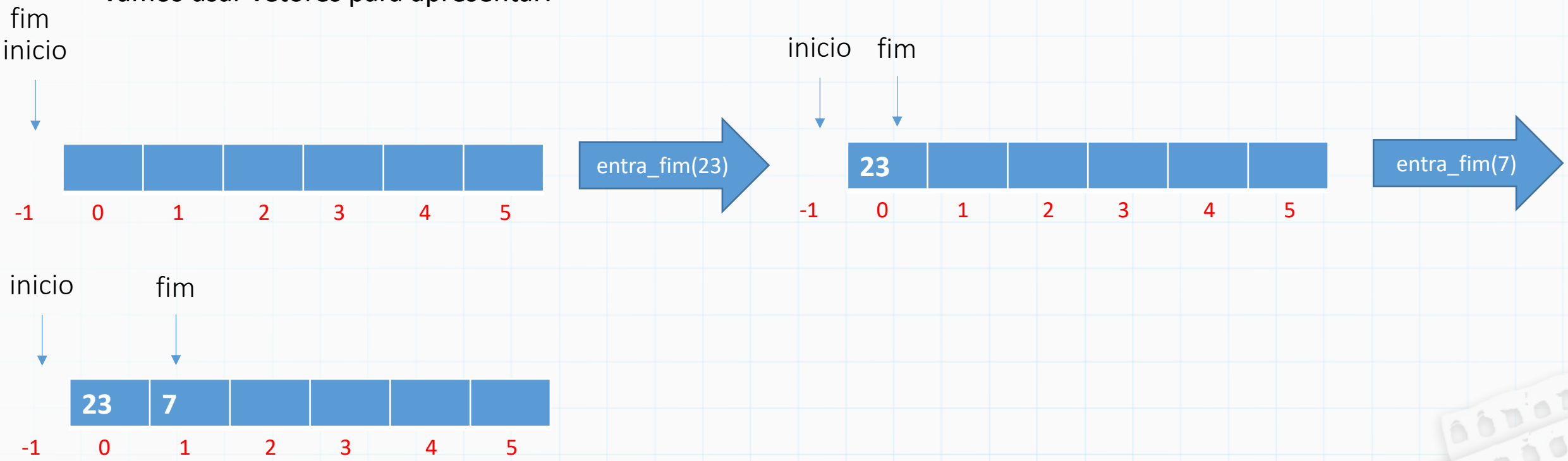
- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



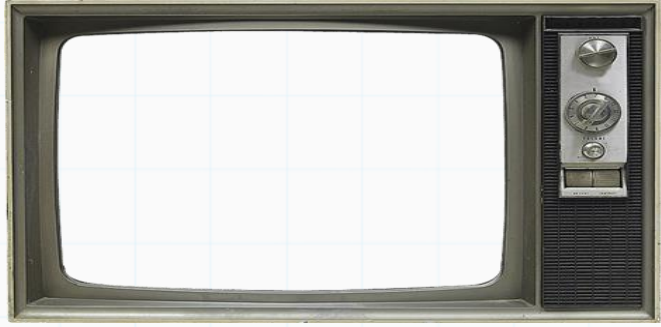
Filas



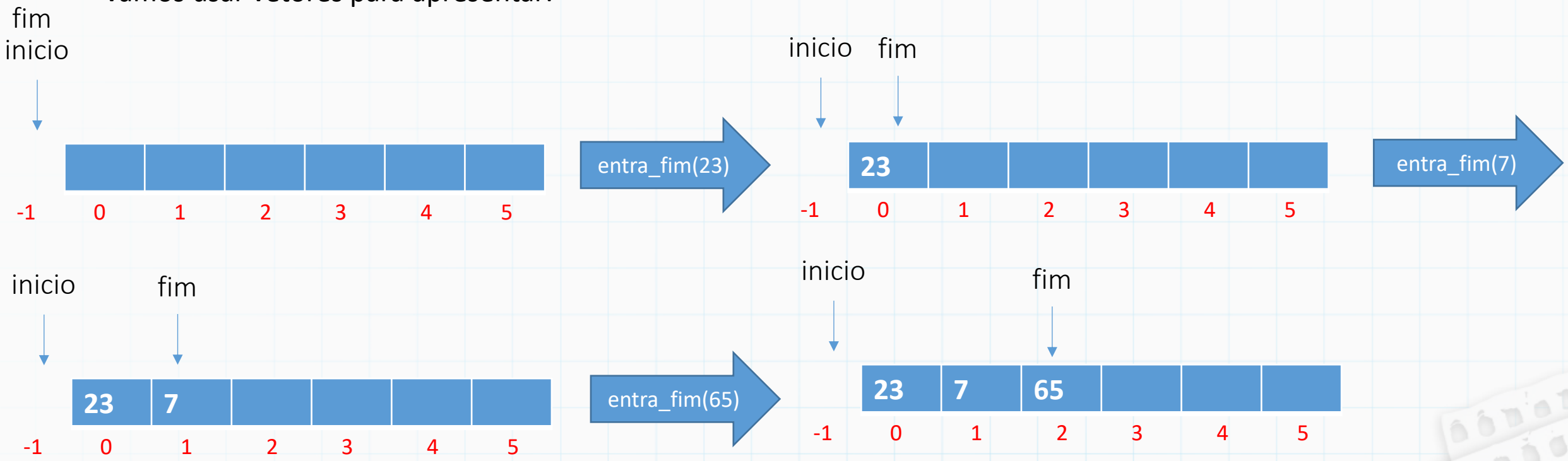
- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



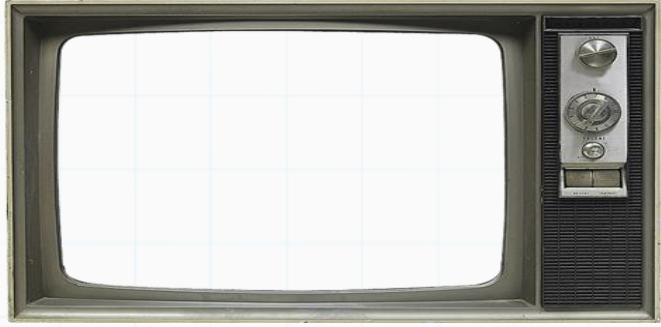
Filas



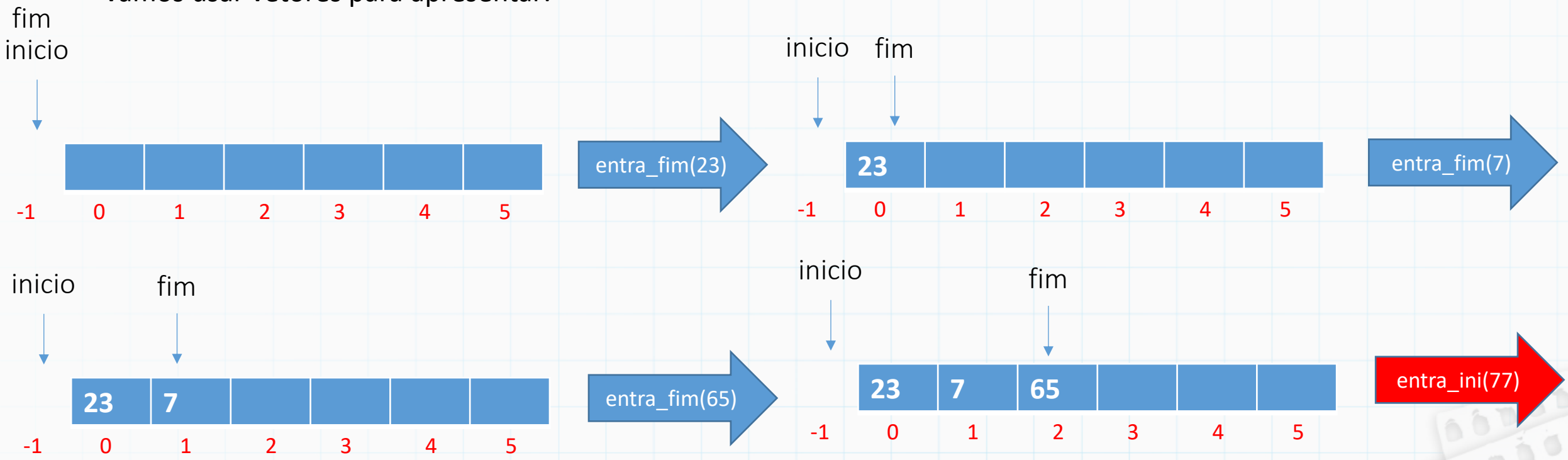
- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



Filas

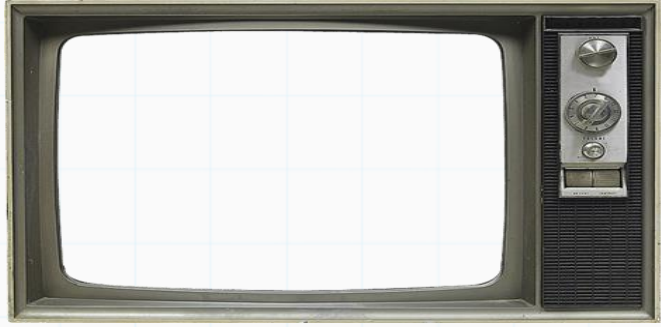


- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:

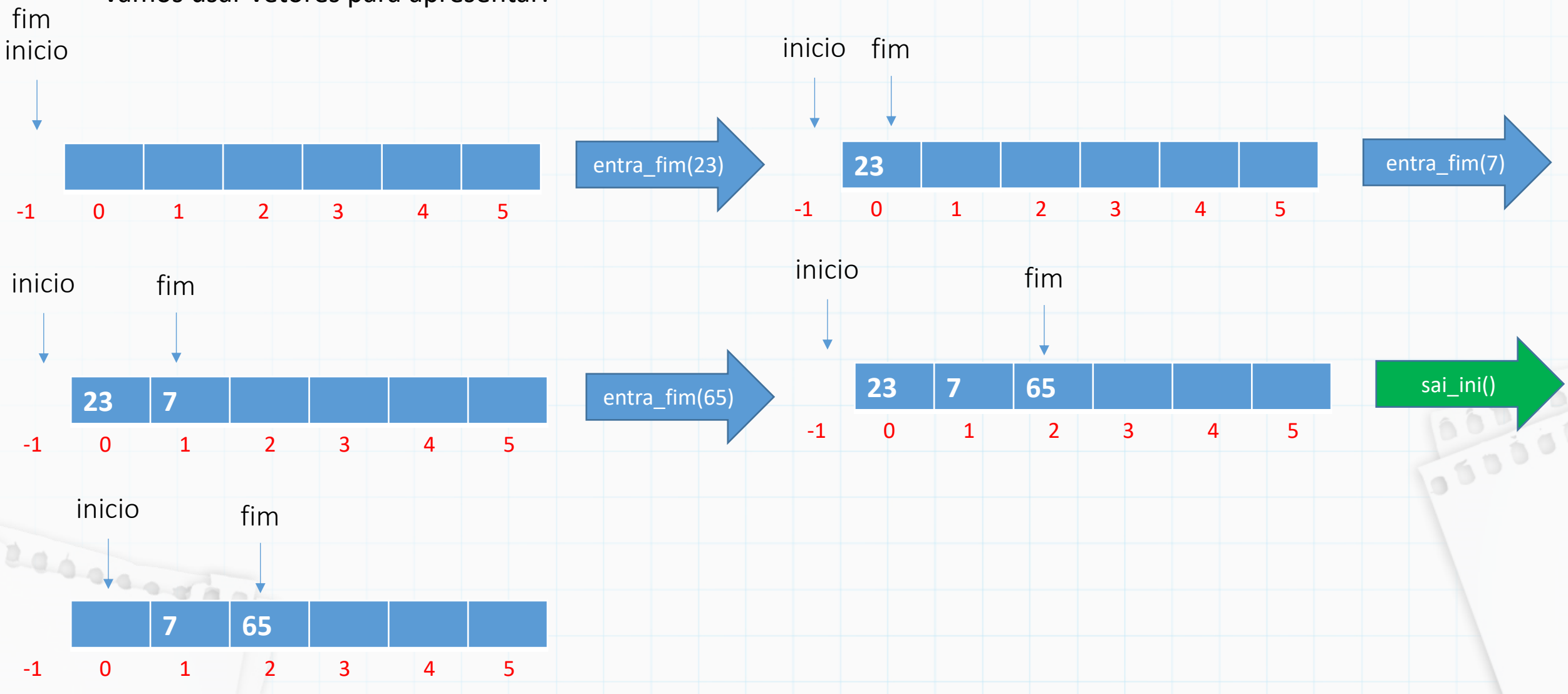


Não tem espaço para entrar no inicio

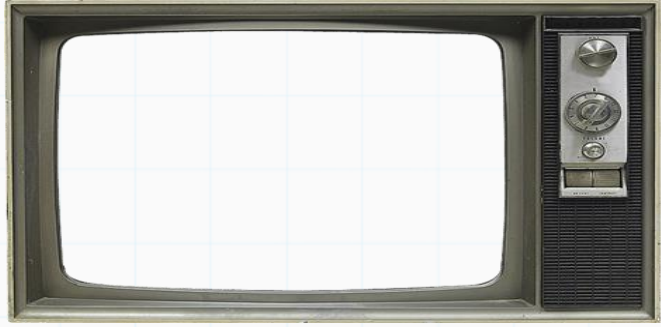
Filas



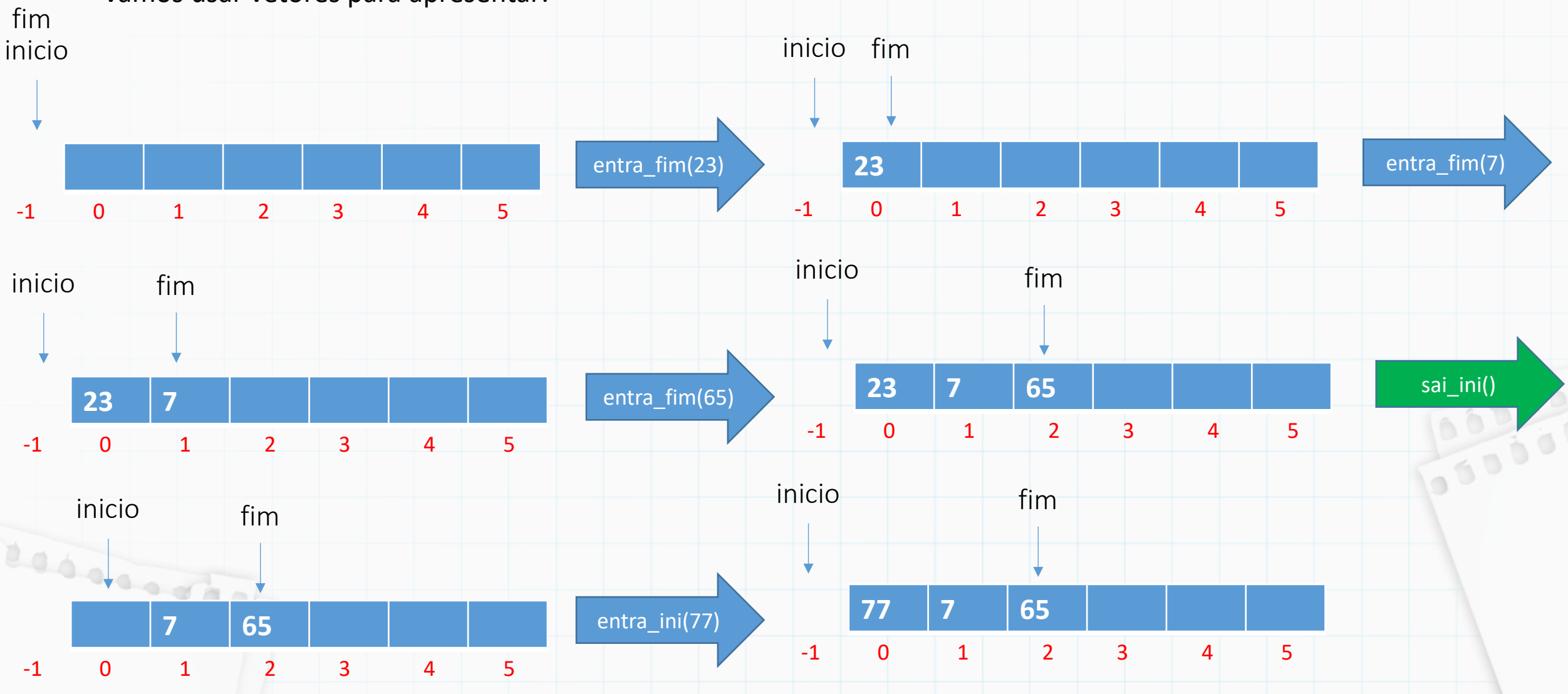
- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



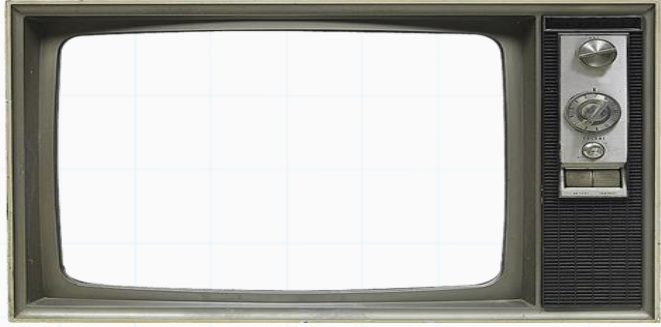
Filas



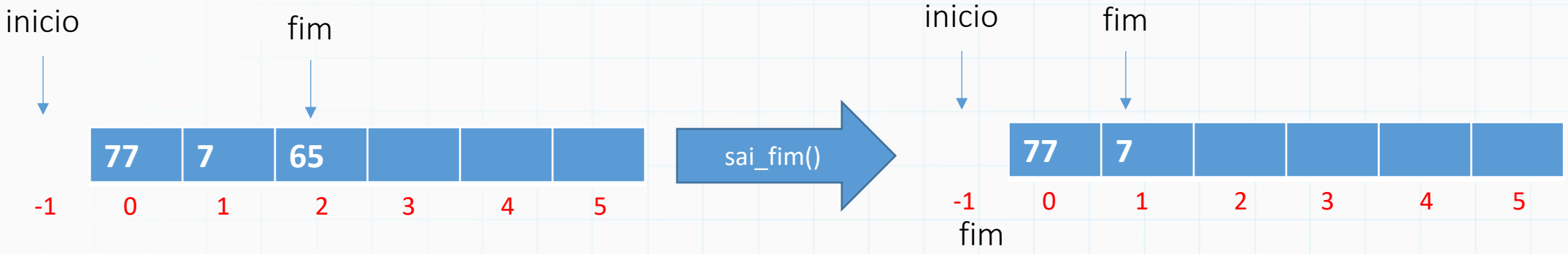
- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



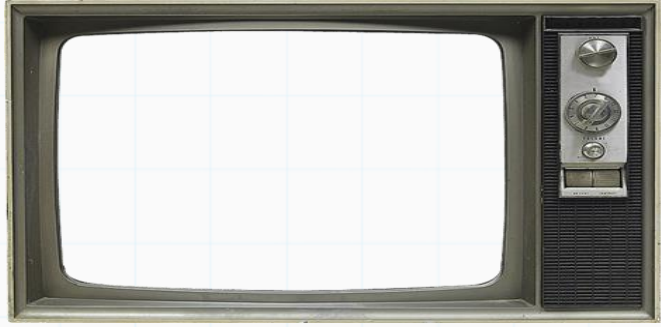
Filas



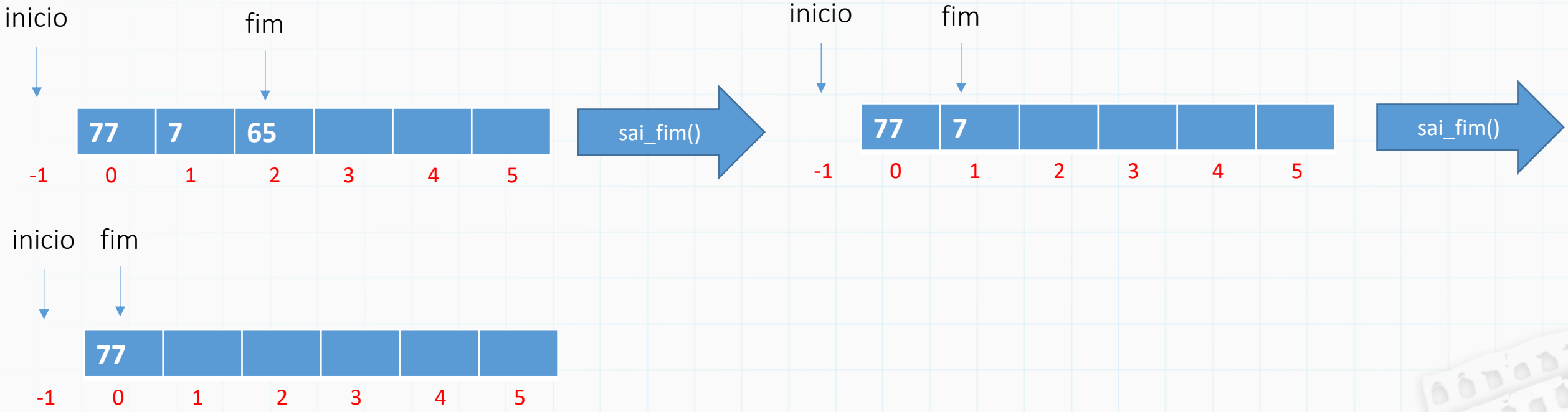
- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



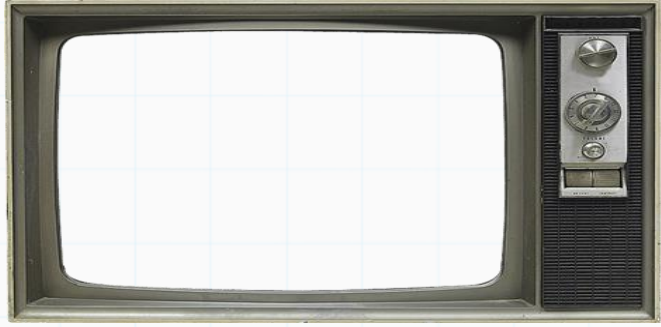
Filas



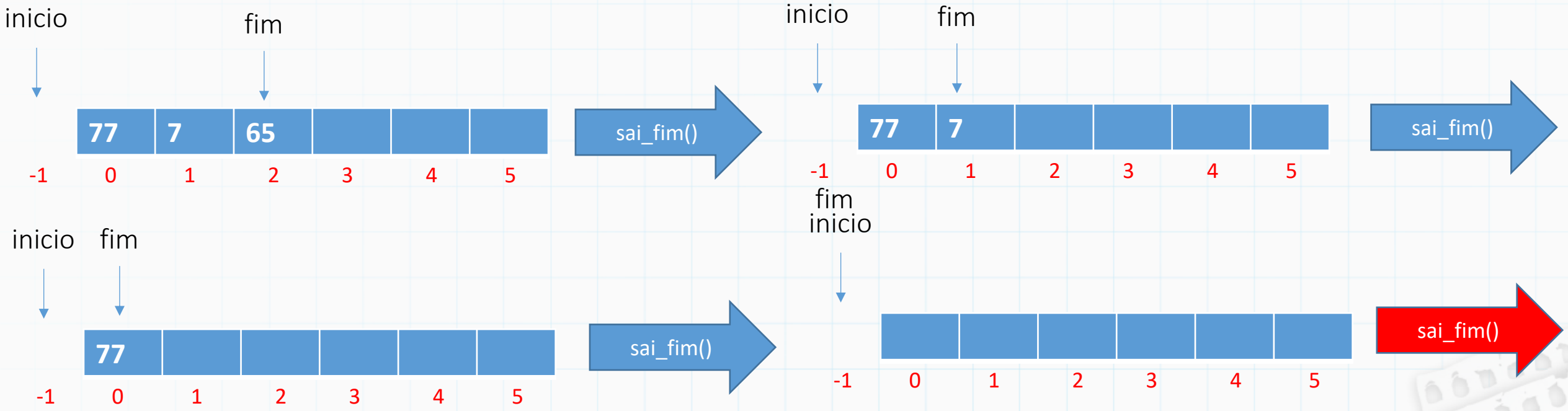
- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



Filas

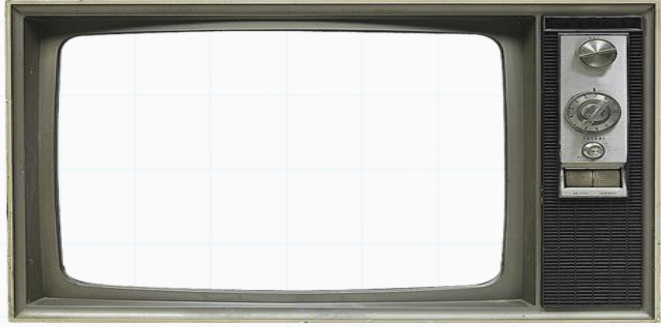


- D.E. (*Double End*) Filas :
  - São filas em que podemos realizar entradas e saídas pelos dois extremos.
  - Podem ser implementadas tanto por vetores quanto por listas encadeadas.
  - Vamos usar vetores para apresentar:



Está vazio

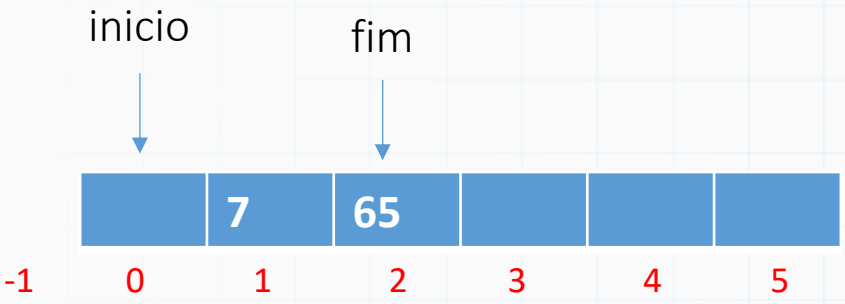
Filas



- Exercício 3) Escreva as funções de entrar_ini, entrar_fim, sair_ini, sair_fim para D.E. (*Double End*) Filas :

```
struct FILA
{
int tam;
int ini;
int fim;
int *v;
};typedef struct FILA fila;
```

```
void cria_fila(fila* F, int tam)
{
    F->ini = -1;
    F->fim = -1;
    F->tam  = tam;
    F->v    = (int*) malloc( F->tam * sizeof(int));
}
```




```
int main()
{
    fila F;
    cria_fila(&F, 7);

    entra_fim_fila(&F, 54);
    entra_fim_fila(&F, 23);
    printf("%d\n", sai_ini_fila(&F));
    entra_ini_fila(&F, 100);
    printf("%d\n", sai_fim_fila(&F));
    …
```

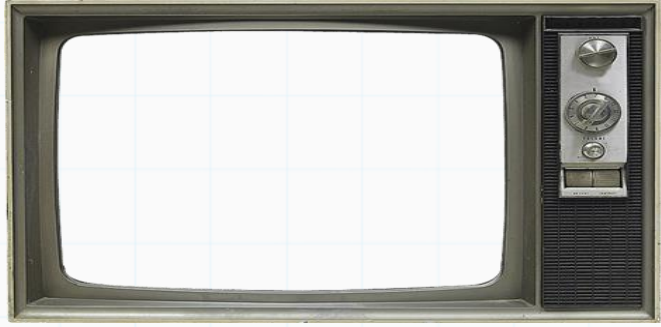
```
int vazia_fila(fila * F)
{
    if (F->ini == F->fim)
        return 1;
    else
        return 0;
}


int cheia_fila(fila * F)
{
    if (F->fim == F->tam-1)
        return 1;
    else
        return 0;
}
```

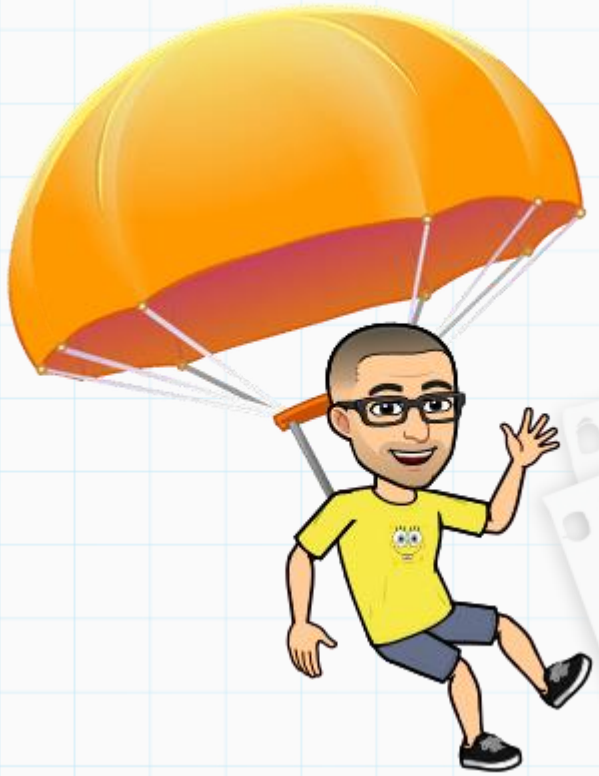
Filas

```c
void entra_ini_fila(fila * F, int el)
{
    if(F->ini == -1)
        printf("fila cheia pela frente\n");
    else
    {
        F->v[F->ini] = el;
        F->ini--;
    }
}


void entra_fim_fila(fila * F, int el)
{
    if(cheia_fila(F) == 1)
        printf("fila cheia por tras\n");
    else
    {
        F->fim++;
        F->v[F->fim] = el;
    }
}
```

```c
int sai_ini_fila(fila * F)
{
    int el;

    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
    {
        F->ini++;
        return F->v[F->ini];
    }

    return -1;
}

int sai_fim_fila(fila * F)
{
    int el;

    if(vazia_fila(F) == 1)
        printf("fila vazia\n");
    else
    {
        el = F->v[F->fim];
        F->fim--;
        return el;
    }

    return -1;
}
```

Até a próxima





Slides baseados no curso de Aline Nascimento