



Algoritmos Computacionais e Estruturas de Dados

Prof. Me. Felipe Borges

Prof. Felipe Borges

Doutorando em Sistemas de Potência – UFMA – Brasil

Mestre em Sistemas de Potência – UFMA – Brasil

MBA em Qualidade e Produtividade – FAENE – Brasil

Graduado em Engenharia Elétrica – IFMA – Brasil

Graduado em Engenharia Elétrica – Fontys – Holanda

Técnico em Eletrotécnica – IFMA – Brasil

Projetos e Instalações Elétricas – Engenharia – Banco do Brasil

Desenvolvimento e Gestão de Projetos – Frencken Engineering BV

Fila: Conceito

Conceito base: O primeiro a entrar é o primeiro a sair. Em Inglês (Fifo): **F**irst **I**n **F**ist **O**ut

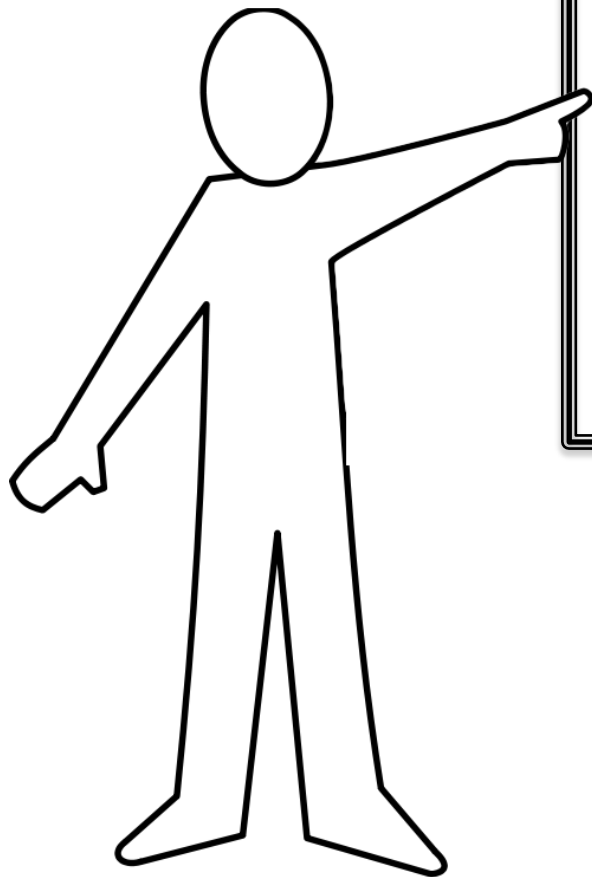


- Fila de banco
- Fila de pacientes
- Fila do supermercado

O Tipo Abstrato Fila

O Tipo Abstrato Fila

Ou seja, as operações que descrevem o comportamento da Fila.



Quais seriam estas
operações, são capazes de
enumerá-las ?

1. criarFila

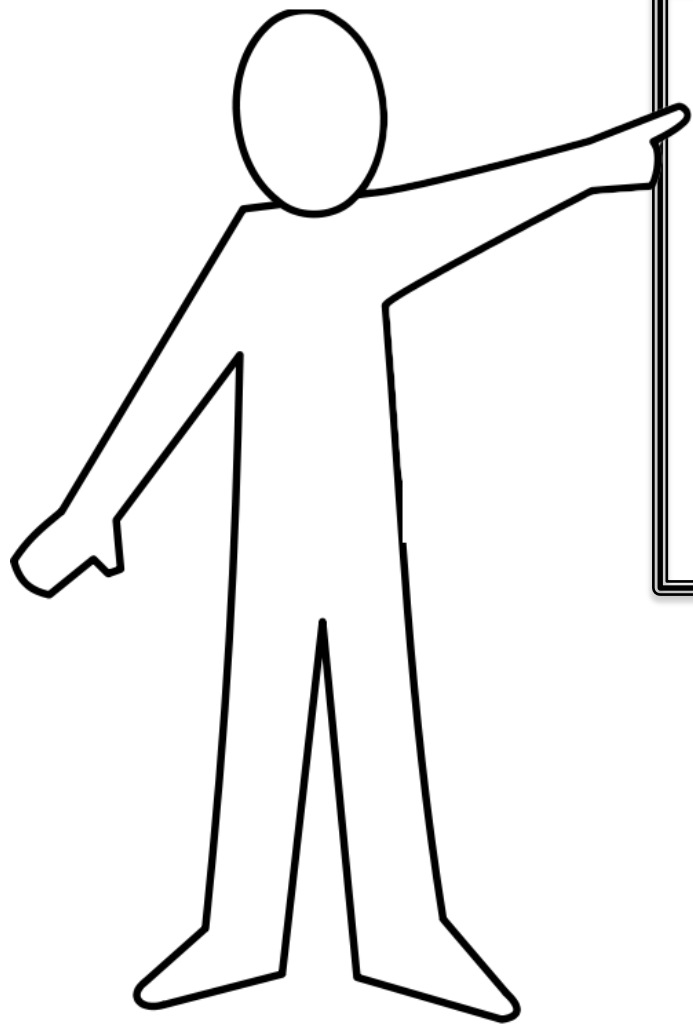
1. criarFila

2. enfileirar

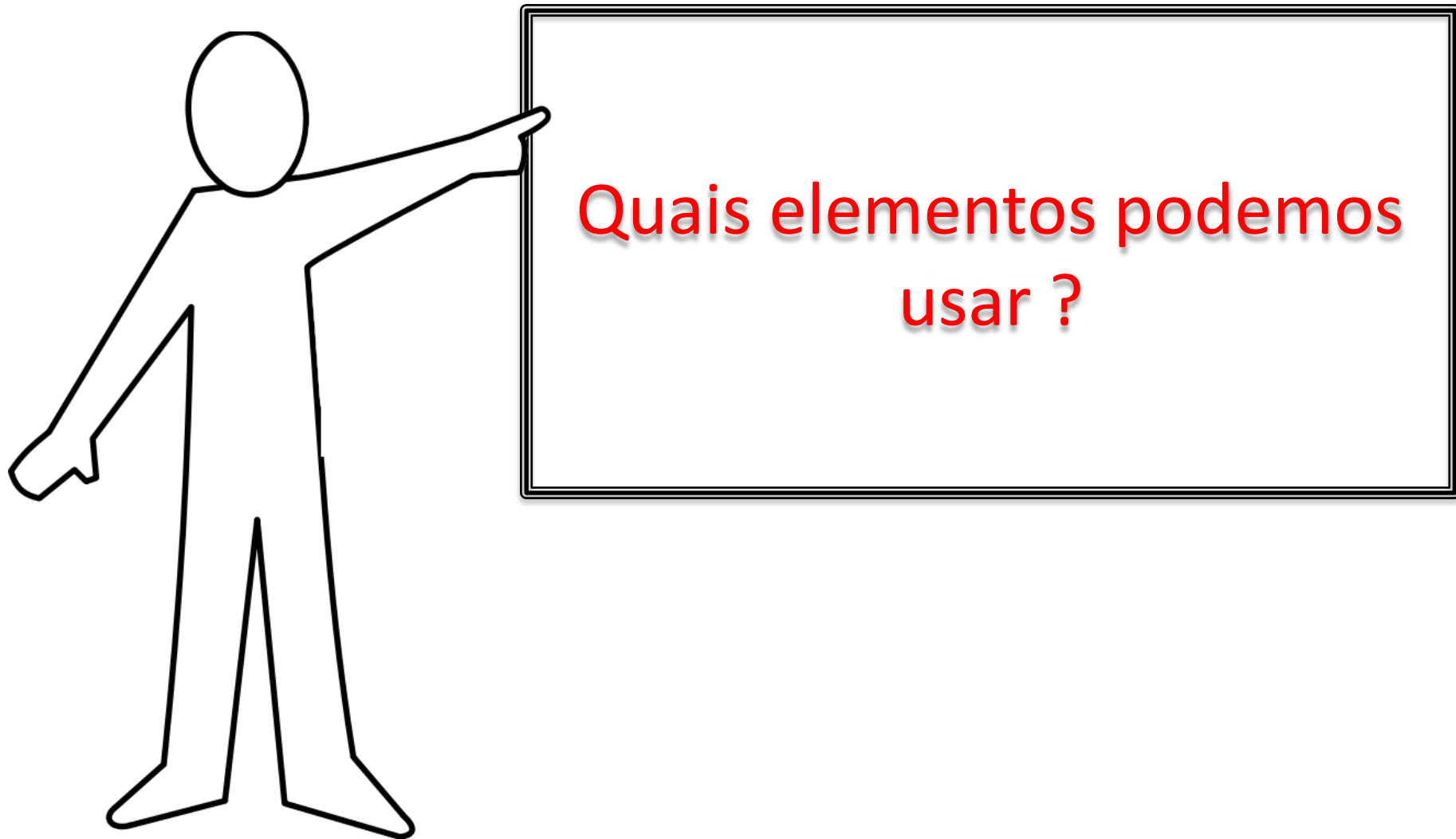
1. criarFila

2. enfileirar

3. desinfileirar



Como podemos codificar
esta estrutura ?



Fila

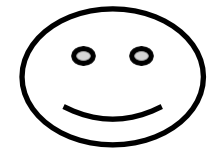
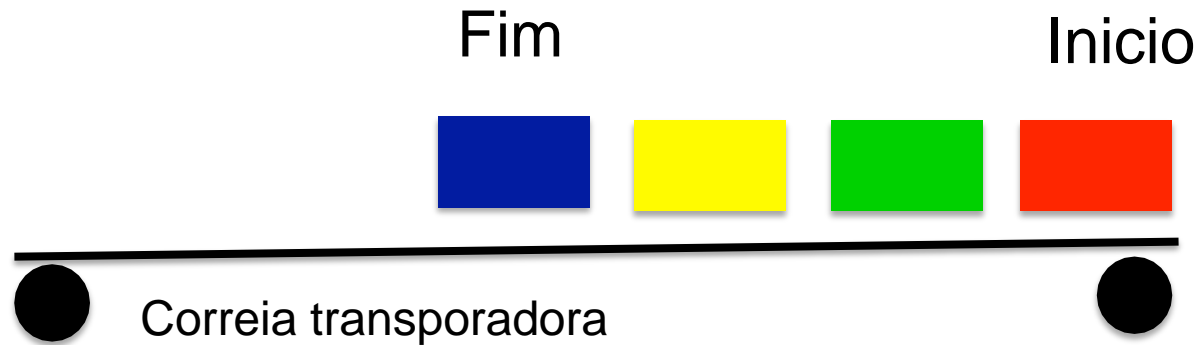
- Novamente, uma possibilidade é usar vetores, dado que estes permitem armazenar uma coleção de dados.
- Uma fila que usa vetor como estrutura básica é chamada de fila estática.
- Na Unidade II, veremos como codificar uma fila dinâmica usando listas encadeadas.



Estava pensando, e observei que na pilha a entrada e saída é pelo mesmo “lado”.
Porém na fila é diferente, quem chega vai para fim da fila e sai quem está no início da fila.

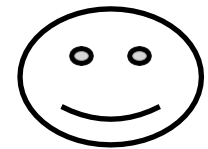
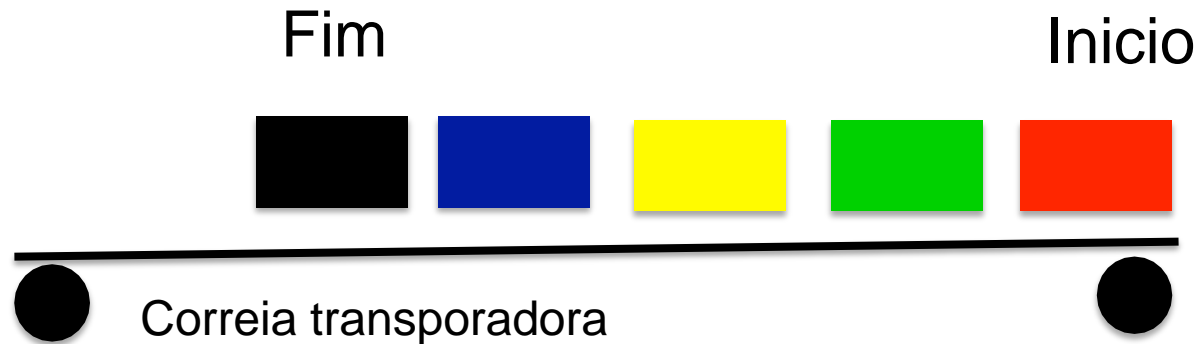


Imaginando, um trabalho em série, onde um conjunto de peças percorre uma correia até ela ir para empacotamento teríamos algo assim:



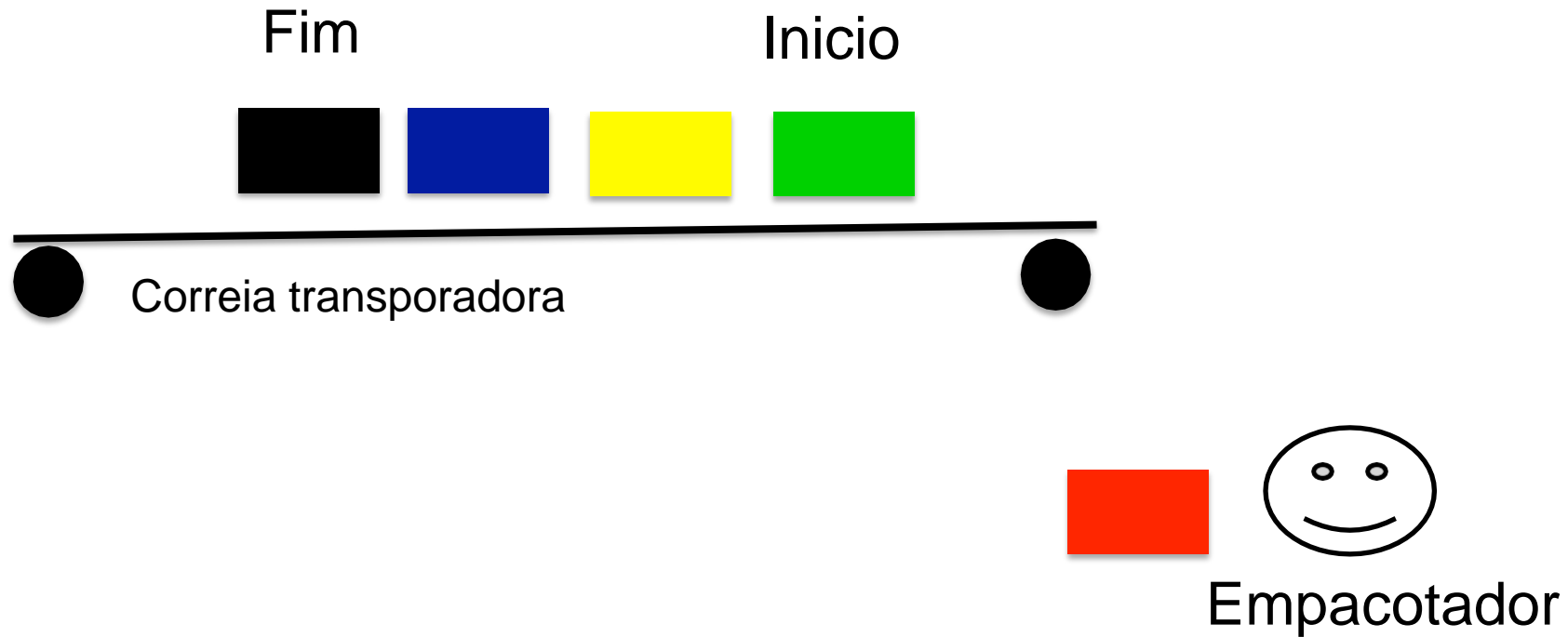
Empacotador

Quando chega um novo produto, ele é colocado no fim da fila.

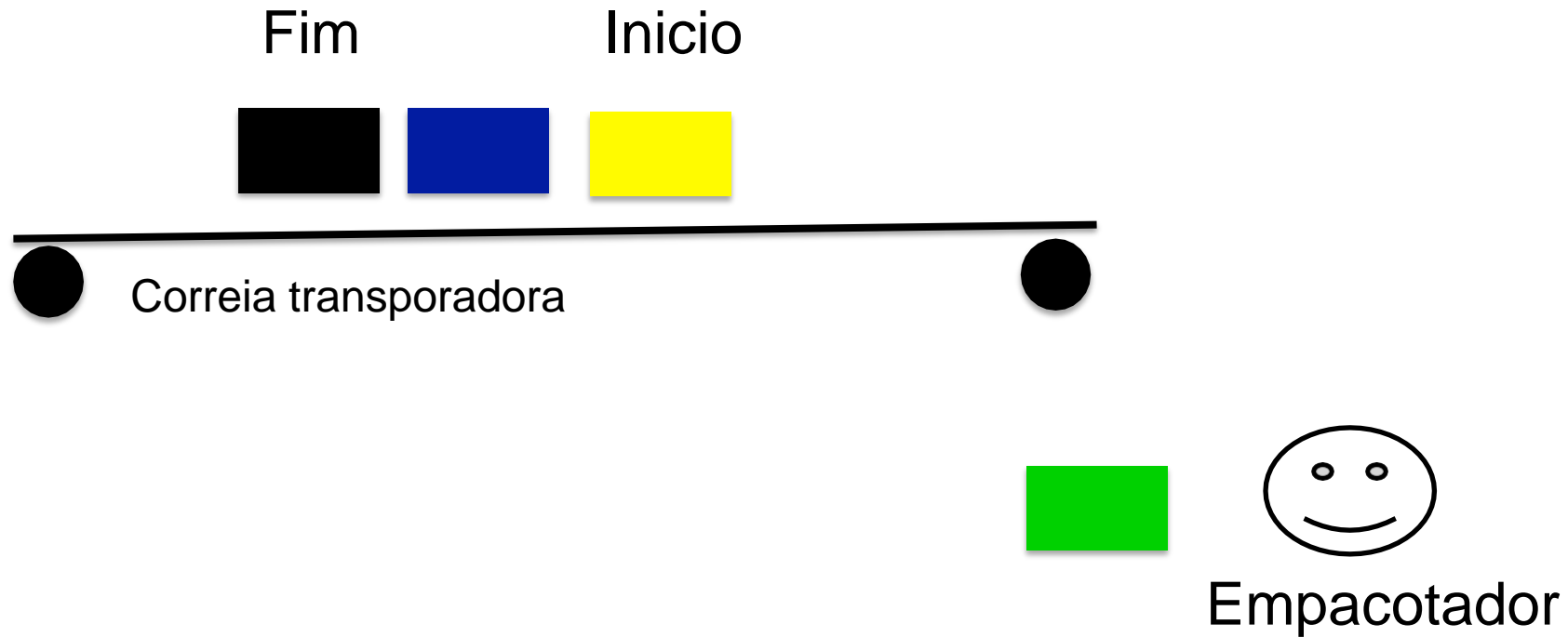


Empacotador

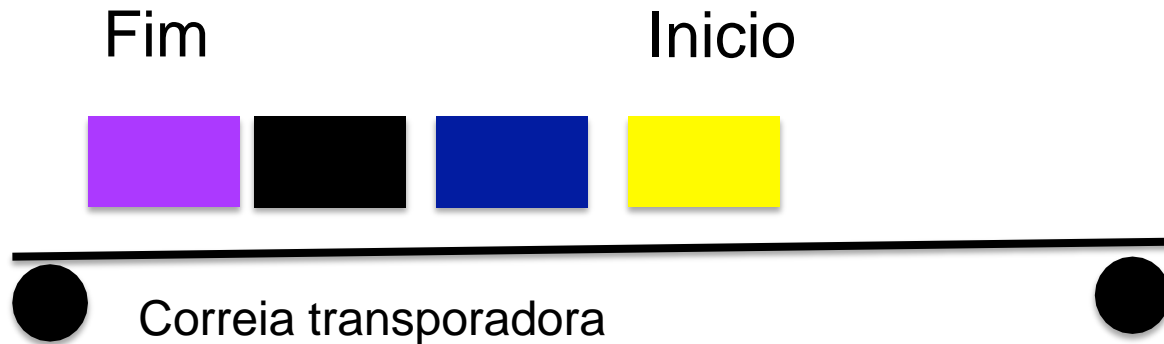
Quando um produto chega ao fim do processo, a peça anterior passa a ser o início da fila.



Quando um produto chega ao fim do processo, a peça anterior passa a ser o início da fila.



Quando chega um novo produto, ele é colocado no fim da fila.



Empacotador

CODIFICANDO ...

TAD: Fila

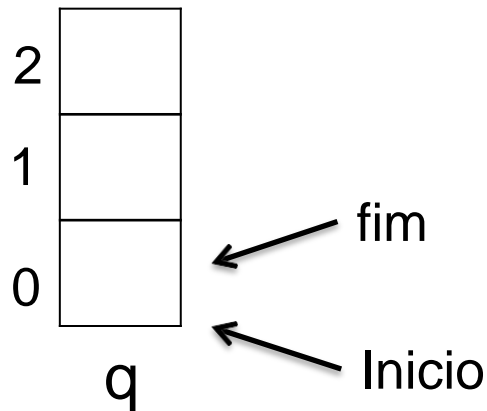
- Definindo o tipo de dados Fila. Observe, que tivemos que incluir os dois atributos, que marca o inicio e o fim da fila.

```
typedef struct {  
    int v[MAX];  
    int inicio, fim;  
} Fila;
```



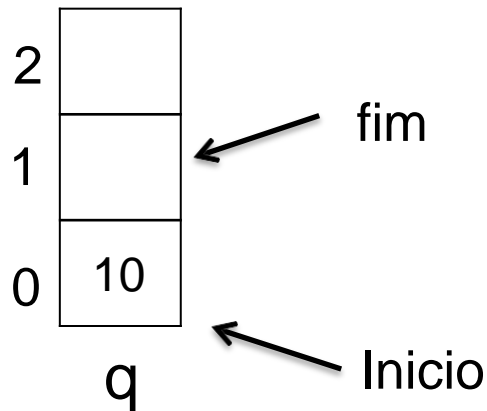
Antes de codificar vou pensar um pouco.
Eu sei que quando eu enfileiro estarei
incrementando o fim e quando eu
desenfileiro eu incremento o inicio. Mas
vou simular antes de codificar.

Considerando uma fila de no máximo 3 elementos:



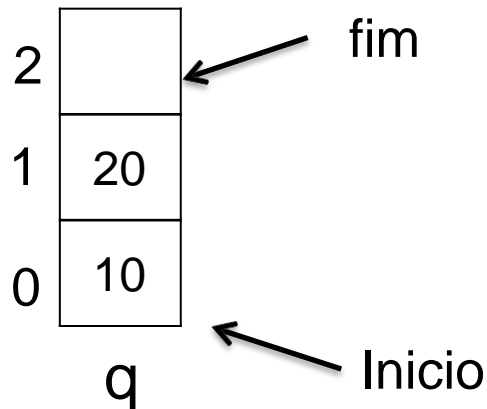
```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

Considerando uma fila de no máximo 3 elementos:



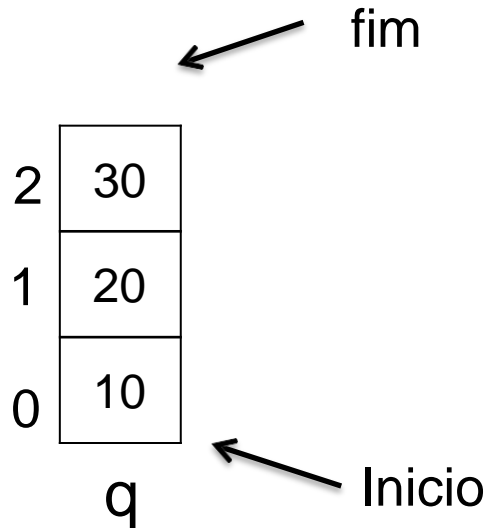
```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```


Considerando uma fila de no máximo 3 elementos:



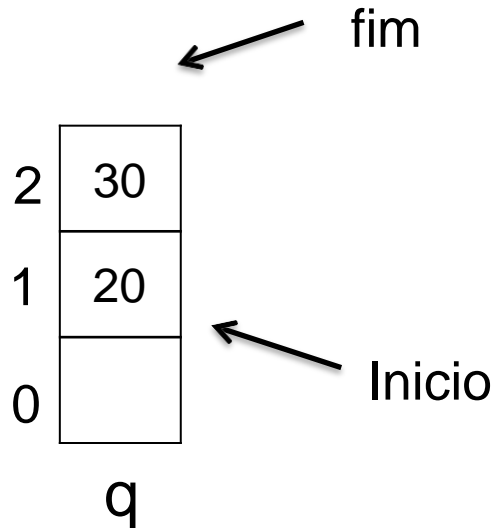
```
Fila *q = criaFila ();
enfileira (q, 10);
enfileira (q, 20);
enfileira (q, 30);
printf ("%d\n", desinfileira(q));
printf ("%d\n", desinfileira(q));
enfileira (q, 50);
enfileira (q, 60);
printf ("%d\n", desinfileira(q));
printf ("%d\n", desinfileira(q));
printf ("%d\n", desinfileira(q));
```

Considerando uma fila de no máximo 3 elementos:



```
Fila *q = criaFila ();
enqueue (q, 10);
enqueue (q, 20);
enqueue (q, 30);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
enqueue (q, 50);
enqueue (q, 60);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
```

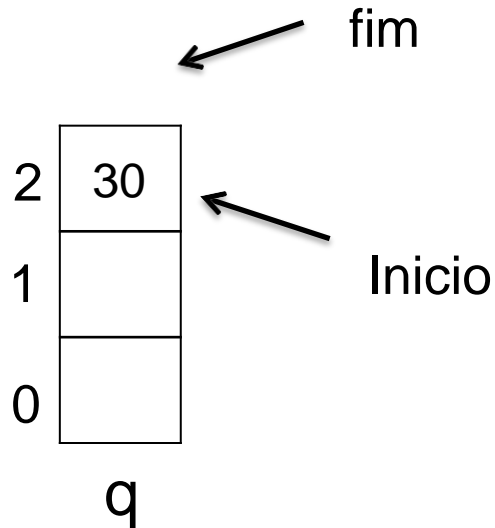
Considerando uma fila de no máximo 3 elementos:



```
Fila *q = criaFila ();
enqueue (q, 10);
enqueue (q, 20);
enqueue (q, 30);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
enqueue (q, 50);
enqueue (q, 60);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
```

10

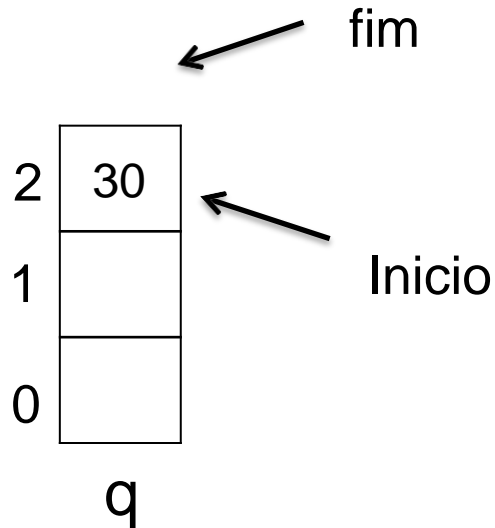
Considerando uma fila de no máximo 3 elementos:



10
20

```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

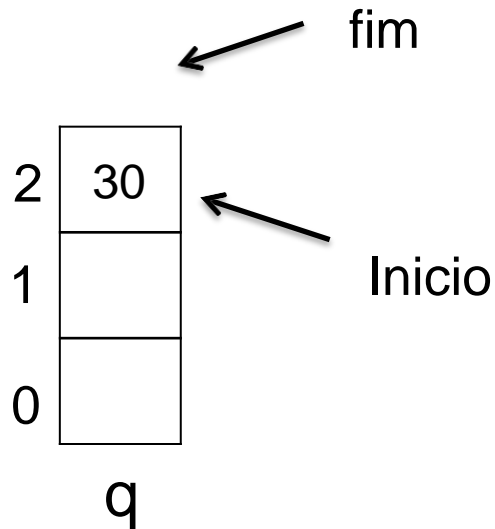
Considerando uma fila de no máximo 3 elementos:



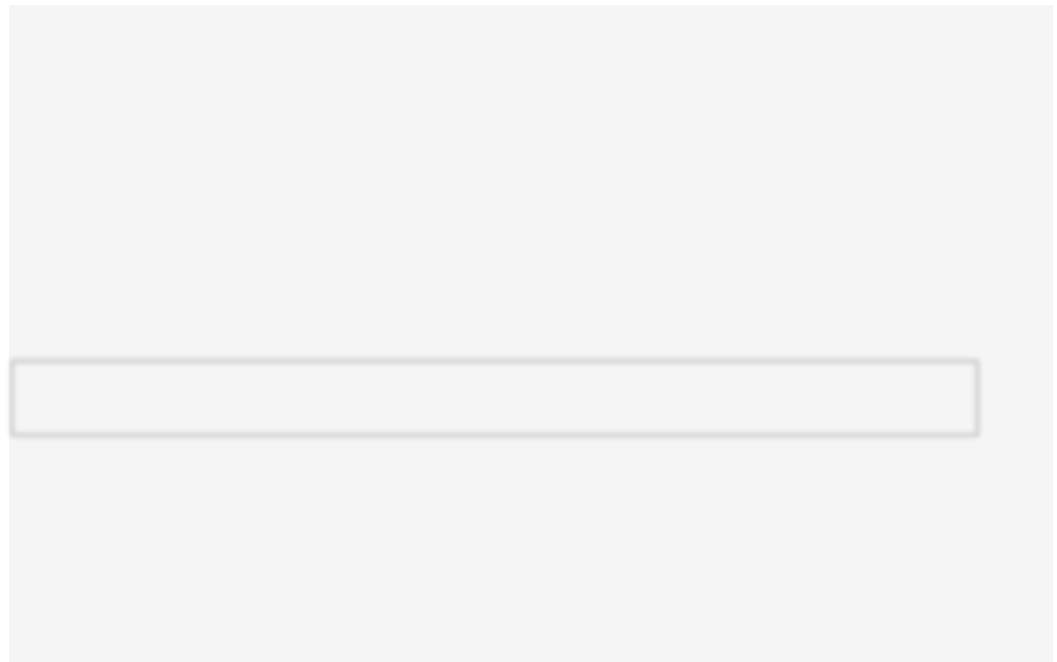
10
20

```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

Considerando uma fila de no máximo 3 elementos:

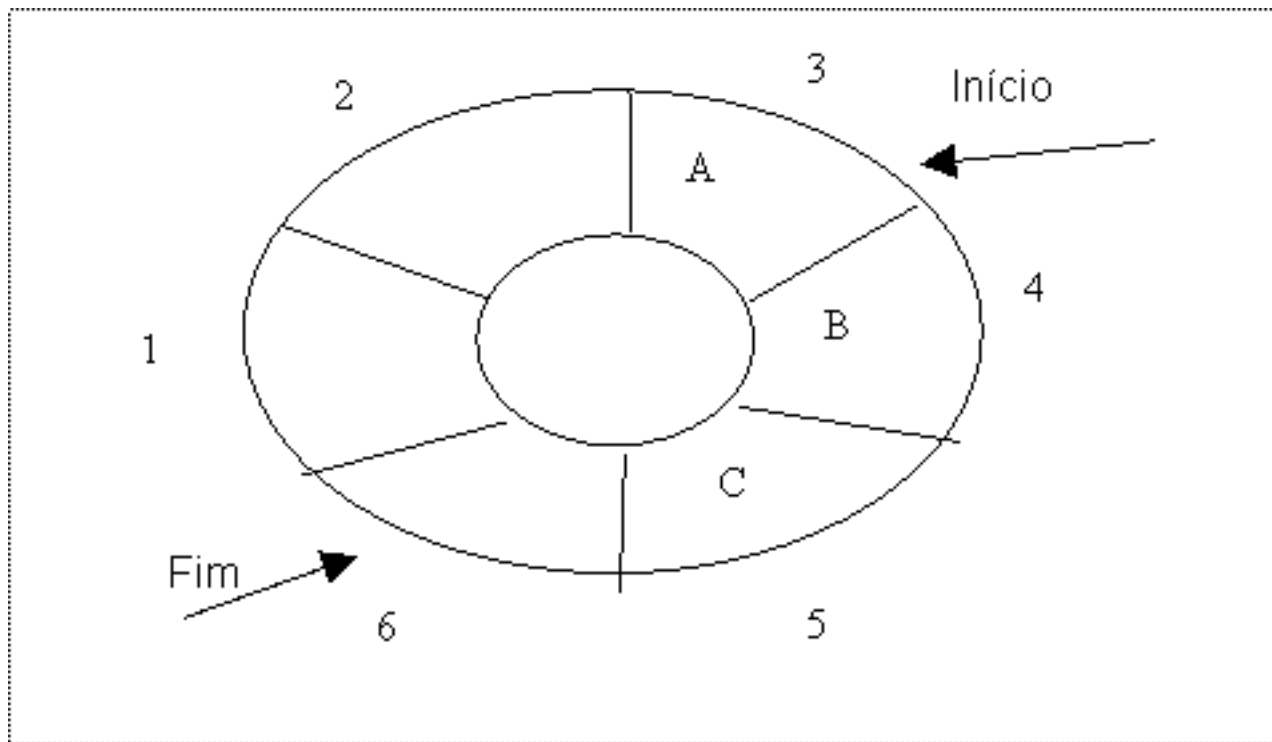


Erro: Mesmo tendo áreas vagas, eu não posso mais adicionar elemento.



TAD: Fila

- Solução, vetor circular:



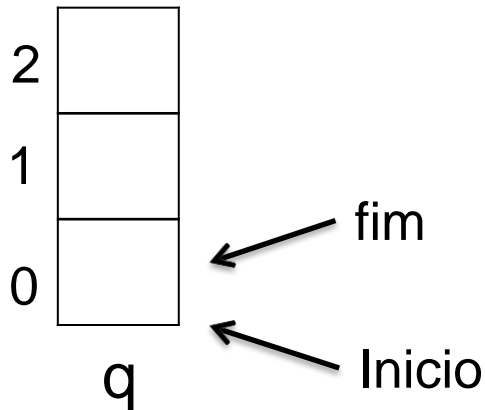
Depois do último elemento, temos o primeiro e antes do primeiro temos o último.

Fonte: <http://www2.dc.ufscar.br/~bsi/materiais/ed/u5.html>



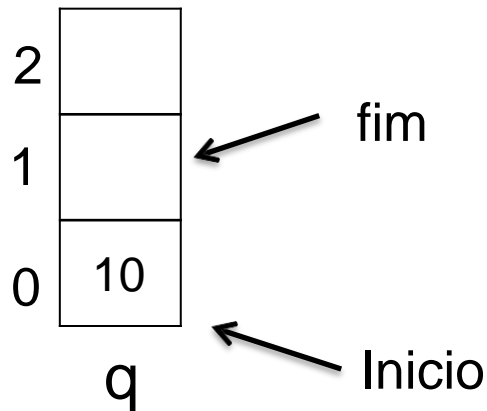
Agora vou simular novamente, mas considerando meu vetor circular.

Considerando uma fila de no máximo 3 elementos:



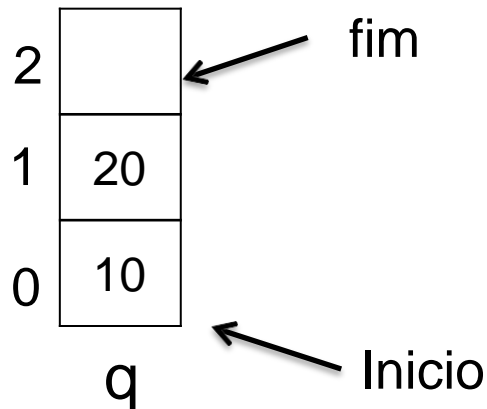
```
Fila *q = criaFila ();
enqueue (q, 10);
enqueue (q, 20);
enqueue (q, 30);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
enqueue (q, 50);
enqueue (q, 60);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
```

Considerando uma fila de no máximo 3 elementos:



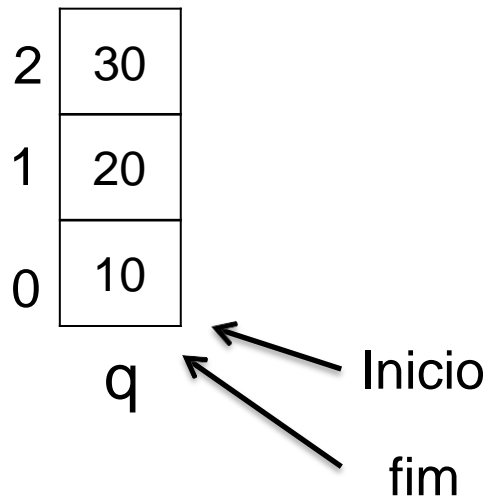
```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

Considerando uma fila de no máximo 3 elementos:



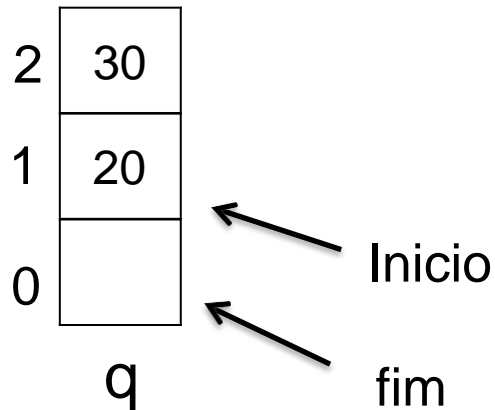
```
Fila *q = criaFila ();
enqueue (q, 10);
enqueue (q, 20);
enqueue (q, 30);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
enqueue (q, 50);
enqueue (q, 60);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
```

Considerando uma fila de no máximo 3 elementos:



```
Fila *q = criaFila ();
enqueue (q, 10);
enqueue (q, 20);
enqueue (q, 30);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
enqueue (q, 50);
enqueue (q, 60);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
```

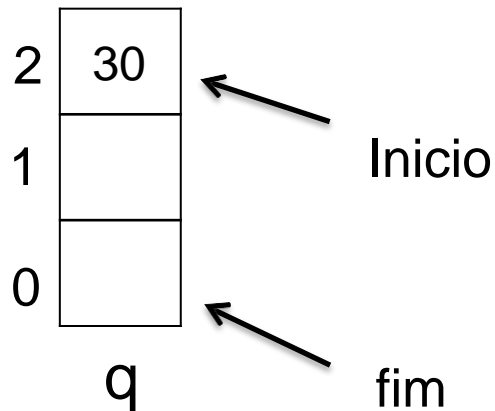
Considerando uma fila de no máximo 3 elementos:



```
Fila *q = criaFila ();
enqueue (q, 10);
enqueue (q, 20);
enqueue (q, 30);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
enqueue (q, 50);
enqueue (q, 60);
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
printf ("%d\n", dequeue(q));
```

10

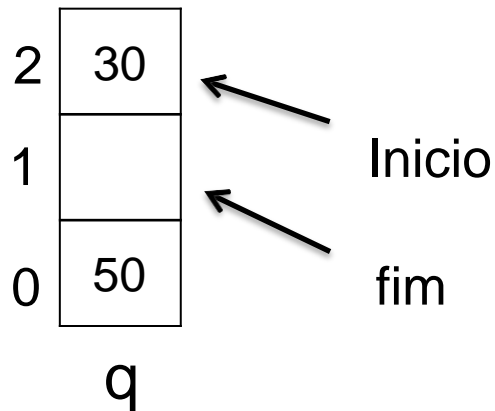
Considerando uma fila de no máximo 3 elementos:



10
20

```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

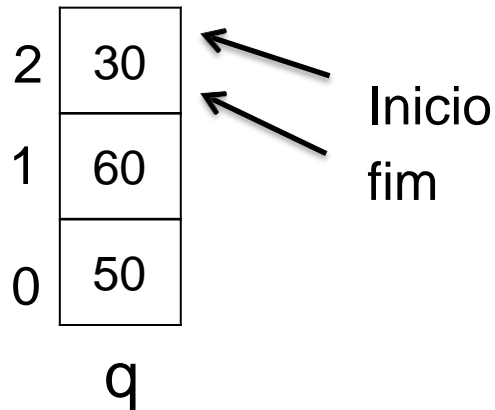
Considerando uma fila de no máximo 3 elementos:



10
20

```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

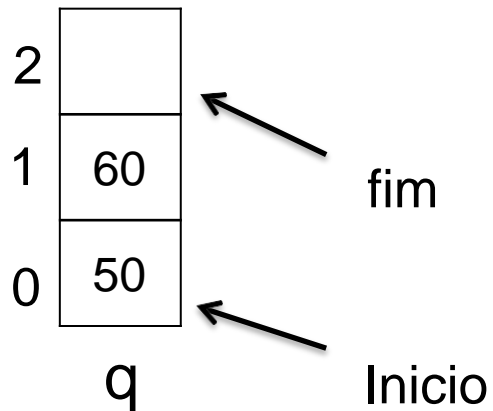
Considerando uma fila de no máximo 3 elementos:



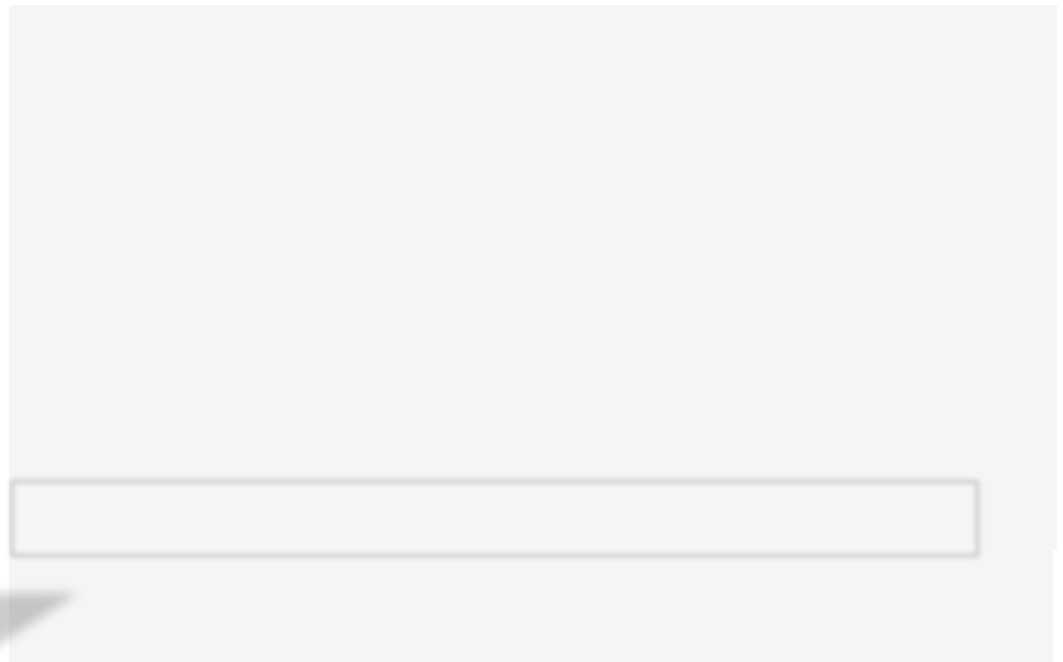
10
20

```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

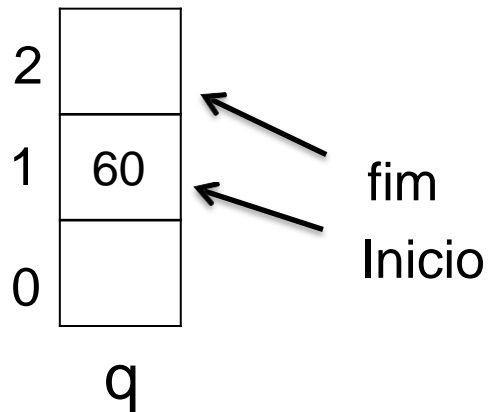

Considerando uma fila de no máximo 3 elementos:



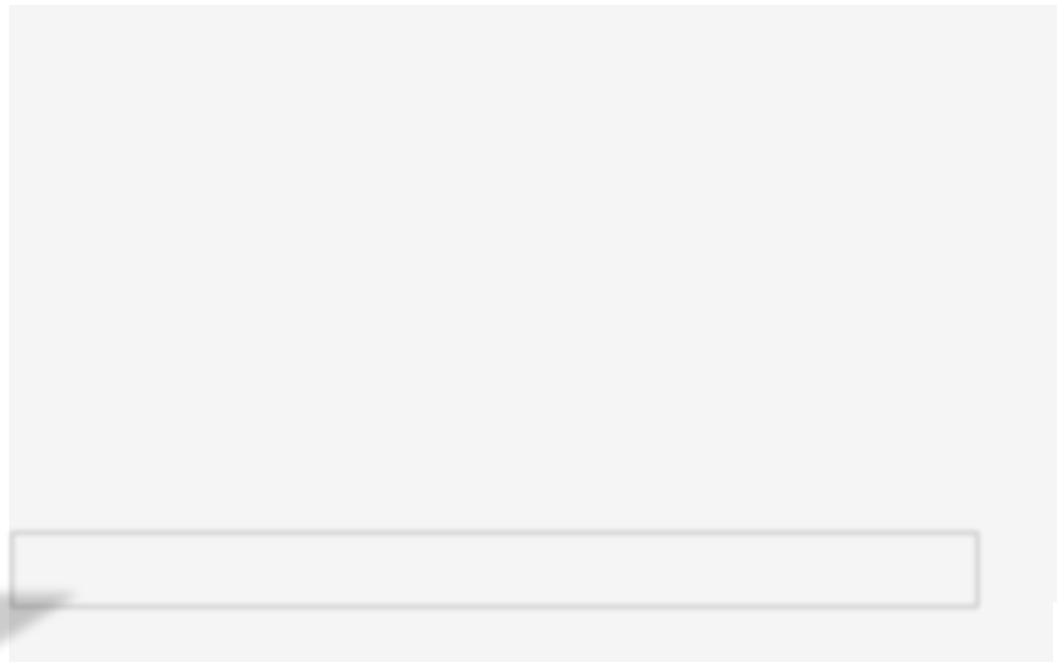
10
20
30



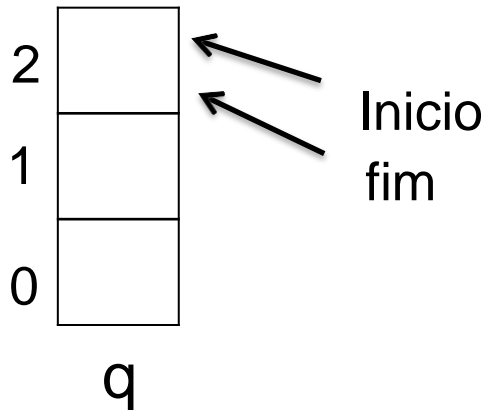
Considerando uma fila de no máximo 3 elementos:



```
10
20
30
50
```



Considerando uma fila de no máximo 3 elementos:




10
20
30
50
60

```
Fila *q = criaFila ();  
enqueue (q, 10);  
enqueue (q, 20);  
enqueue (q, 30);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
enqueue (q, 50);  
enqueue (q, 60);  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));  
printf ("%d\n", dequeue(q));
```

TAD: Fila

- Testando nossa codificação:

```
int main () {  
    Fila *q = criaFila ();  
    enfileira (q, 10);  
    enfileira (q, 20);  
    enfileira (q, 30);  
    printf ("%d\n", desinfileira(q));  
    printf ("%d\n", desinfileira(q));  
    enfileira (q, 40);  
    printf ("%d\n", desinfileira(q));  
    printf ("%d\n", desinfileira(q));  
    return 0;  
}
```



O que será
impresso ?

Exemplos de Aplicação

Fila: Aplicações

- Em controle de processos no sistema operacional, fila de processos.
- Em algoritmo de busca em grafo (busca em largura).
- Algoritmo de caminho mínimo (Dijkstra).
- Algoritmo de ordenação.

Fila: Aplicações

- Filas de tarefas no sistema operacional
 - Cada programa de computador consiste em um conjunto de processos que requisita recursos (processamento, acesso à hardware, etc).
 - O sistema operacional enfileira os processos e vai executando cada um, seguindo diferentes regras e prioridades
- Filas de mensagens a serem transmitidas em uma rede
 - Diversos programas do PC trocando dados na internet e utilizando o mesmo hardware de rede
- Filas de lotes de produtos a serem fabricados em uma linha de produção
- Etapas de produção em um sistema
- Filas de compra de ingressos em sites online

Referências Bibliográficas

- AARON, Tanenbaun. **Estruturas de Dados usando C.** São Paulo: Makron Books, 1995.
- PEREIRA, Silvio. **Estruturas de Dados Fundamentais.** São Paulo: Editora Erica, 1996.
- VELOSO, Paulo et al. **Estruturas de Dados.** Rio de Janeiro: Editora Campus, 1983.

Fila (Queue)

- Representa um tipo de estrutura onde novos elementos são adicionados ao final da fila
 - FIFO – First In, First Out
 - O primeiro a entrar é o primeiro a sair

Início da Fila

Fim da Fila



Fila (queue)



- Características

- Número de elementos - n
 - Máximo número de elementos que cabe na fila
- Última posição - *fim*
 - Posição onde um novo elemento entrará

$n = 5$
 $fim = 3$

- Operações

- Inserir
 - Adiciona um novo elemento ao fim da fila
 - Incrementa *fim*
 - Queue overflow (fila cheia) se $fim = n$
- Remover
 - Remover o elemento à frente da fila
 - Desloca os elementos (se for um vetor)
 - Decrementa o *fim*
 - Stack empty (fila vazia) se $fim = 0$

Fila (Queue)

- Pode ser implementada com vetor
- Variáveis de controle da fila
 - Número de elementos - n
 - Máximo número de elementos que cabe na fila
 - Última posição - *fim*
 - Posição onde um novo elemento entrará na fila
- A cada remoção, o conteúdo da posição inicial deve ser substituído
 - $Fila[0] = fila[1]$
 - $Fila[1] = fila[2]$
 - Até o *fim* da fila – com um laço for é possível implementar

Fila (Queue)

Um exemplo de struct para uma fila

```
typedef struct {  
    int fila[4];    // tamanho da fila  
    int fim;  
} Fila;
```

fim = 0



Fila (Queue)

```
#include <stdio.h>
#include <stdlib.h>
#define ELEMENTOS_FILA 4
typedef struct{
    int vetor[ELEMENTOS_FILA];
    int fim
} Fila;
int main()
{
    Fila f;
    f.fim = 0; //nao tem ngm na fila inicialmente

    //incluir um numero na fila
    f.vetor[f.fim] = 15;
    f.fim ++;

    //incluir mais um numero na fila
    f.vetor[f.fim] = 25;
    f.fim ++;

    return 0;
}
```

Inserindo 2 itens na FILA

Fila (Queue)

```
int main()
{
    Fila f;
    f.fim = 0; //nao tem ngm na fila inicialmente
    //incluir um numero na fila
    f.vetor[f.fim] = 15;
    f.fim ++;
    //incluir mais um numero na fila
    f.vetor[f.fim] = 25;
    f.fim ++;

    //retirar
    printf("Elemento que sai da Fila: %d. \n",f.vetor[0]);
    //deslocamento na fila
    f.vetor[0]=f.vetor[1];
    f.vetor[1]=f.vetor[2];
    f.fim --; //ultima posição recuou 1

    return 0;
}
```

Retirando 2 itens da FILA

Fila (Queue)

```
int main()
{
    int i;
    Fila f;
    f.fim = 0; //nao tem ngm na fila inicialmente
    //incluir um numero na fila
    f.vetor[f.fim] = 15;
    f.fim ++;

    //incluir mais um numero na fila
    f.vetor[f.fim] = 15;
    f.fim ++;

    //retirar
    printf("Elemento que sai da Fila: %d. \n",f.vetor[0]);
    //deslocamento na fila
    for(i=0;i<(f.fim-1);i++){
        f.vetor[i]=f.vetor[i+1];
    }
    f.fim --; //ultima posição recuou 1
    return 0;
}
```

**Retirando 2 itens da FILA
utilizando estrutura de Repetição**

Fila (Queue)

```
int main()
{
    int i;
    Fila f;
    f.fim = 0; //nao tem ngm na fila inicialmente
    //incluir um numero na fila
    f.vetor[f.fim] = 15;
    f.fim++;
    //incluir mais um numero na fila
    f.vetor[f.fim] = 25;
    f.fim++;
    //incluir mais um numero na fila
    f.vetor[f.fim] = 35;
    f.fim++;
    //imprime a fila na tela
    for(i=0;i<f.fim;i++){
        printf("%02d \n",f.vetor[i]);
    }

    //retirar
    printf("Elemento que sai da Fila: %d.
    \n",f.vetor[0]);
    //deslocamento na fila
    for(i=0;i<(f.fim-i);i++){
        f.vetor[i]=f.vetor[i+1];
    }
    f.fim--;
    //imprime a fila na tela
    for(i=0;i<f.fim;i++){
        printf("%02d \n",f.vetor[i]);
    }
    return 0;
}
```

**Imprimindo antes e após retirar 2 itens da FILA
utilizando estrutura de Repetição**

Exercício - Fila

- Faça um programa que implemente uma fila de 20 elementos do tipo inteiro utilizando struct
 - Implemente um método que insira na fila um novo inteiro
 - O método deve enviar uma mensagem se a fila estiver cheia
 - `void adiciona(int valor, Fila *fila);`
 - Implemente um método que remova o primeiro elemento da fila
 - O método deve exibir uma mensagem se a fila estiver vazia
 - `void retira(Fila *fila);`
 - Implemente um método que retorne 1 se a fila está cheia e 0 se não
 - `int isCheia(Fila *fila);`
 - Implemente um método que retorne 1 se a fila está vazia e 0 se não
 - `int isVazia(Fila *fila);`