Filas
Queue

queue
/kyo͞o/

Prof. Bruno Travençolo

Filas

- Estrutura de dados linear usada para armazenar e organizar dados
- Sequência de elementos do mesmo tipo
- Nessa estrutura somente temos conhecimento do primeiro elemento inserido (quem está na "frente" da fila)
  - Diferente de listas, que podemos percorrer todos os elementos.
  - Diferente de listas duplamente encadeadas, que podemos acessar o primeiro/último elemento via funções especializadas nessas tarefas (front()/back()).
- A remoção de um elemento sempre será do elemento que estiver a *mais* tempo na fila
  - Diferente de listas, que podemos remover elementos das extremidades por meio de funções especializadas (pop_front(); pop_back()) ou qualquer elemento erase(int pos)
- É um tipo especial de lista, em que a inserção e a remoção são realizadas sempre em extremidades distintas
- FIFO – First In First Out (primeiro a entrar, primeiro a sair)

Filas

- “Mas como faço então pra consultar um elemento ‘no meio’ da fila?”
  - Não faz. Se for necessário consultar algum elemento diferente do último inserido, isso significa que não é necessário usar uma fila. Talvez uma lista serviria
- Vantagens: estrutura mais simples e mais flexível de implementar. Não há preocupação com vários ponteiros e também o número de operações é menor que uma lista

Filas

- Operações de manipulação da fila
  - Inserir um elemento na fila*
  - Remover um elemento da fila**
  - Acesso ao elemento da fila***
  - Verificar se a fila está cheia ou vazia

- Operações relacionadas a estrutura da fila
  - Criar a fila
  - Destruir a fila

* Inserção sempre no fim da fila

** Remoção sempre no início da fila

*** Acesso sempre ao elemento do início da fila

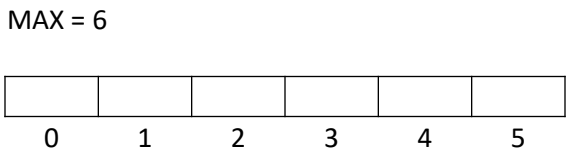
Fila Sequencial Estática

- Será alocado um vetor para armazenar os elementos da fila (`data`)
- Haverá um ponteiro (ou índice) para o início da fila (`front`)
- Haverá um ponteiro (ou índice) para a próxima posição disponível (`rear`)
- Uma variável para indicar a quantidade de elementos na fila (`size`)

```c
#include "TQueue.h"

struct queue{
  int front;
  int rear;
  int size;
  struct aluno data[MAX];
};
```

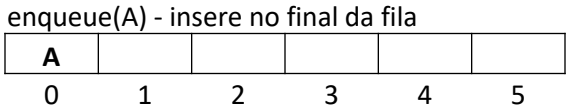
Fila Sequencial Estática

- Insere um elemento na fila
- Incialmente vazia



Fila Sequencial Estática

- Insere um elemento na fila
- Insere "A"



Fila Sequencial Estática

- Insere um elemento na fila
- Insere “A”
- Insere “B”

enqueue(A) - insere no final da fila

A	B				
0	1	2	3	4	5

Fila Sequencial Estática

- Insere um elemento na fila
- Insere "A"
- Insere "B"
- Insere "C"

enqueue(A) - insere no final da fila

A	B	C			
0	1	2	3	4	5

Fila Sequencial Estática

- Remover um elemento
- Quem deve ser removido?

enqueue(A) - insere no final da fila

A	B	C			
0	1	2	3	4	5

Fila Sequencial Estática

- Remover um elemento
- Quem deve ser removido?
  - Elemento "A", pois ele foi o primeiro a entrar na fila
- Como remover?

enqueue(A) - insere no final da fila

A	B	C			
0	1	2	3	4	5

Fila Sequencial Estática

- Como remover?
  - Deslocar todos os elementos

```
for i=0:size-1
  data[i] = data[i+1]
```

dequeue – Remove da fila

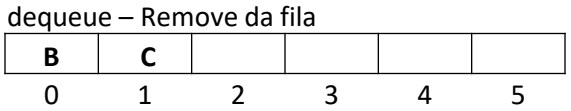
B	C				
0	1	2	3	4	5

  - Problema?

Fila Sequencial Estática

- Como remover?
  - Deslocar todos os elementos

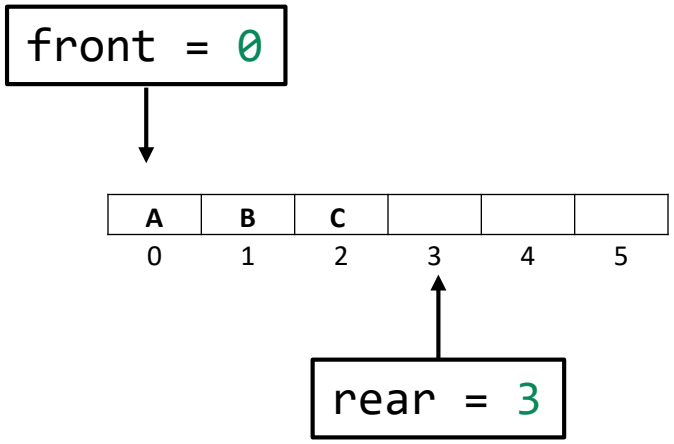
```
for i=0:size-1
  data[i] = data[i+1]
```



  - Problema?
    - Alto custo computacional

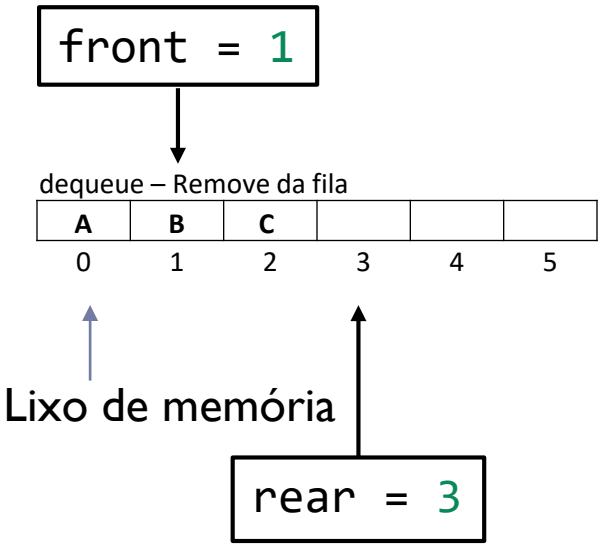
Fila Sequencial Estática

- Como remover?
  - Modificar ponteiro `front`



Fila Sequencial Estática

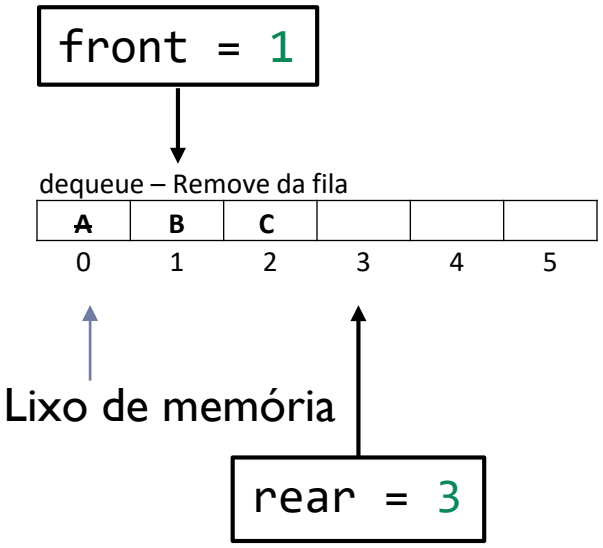
- Como remover?
  - Modificar ponteiro `front`
    - `Remove()`



  - `Problema?`

Fila Sequencial Estática

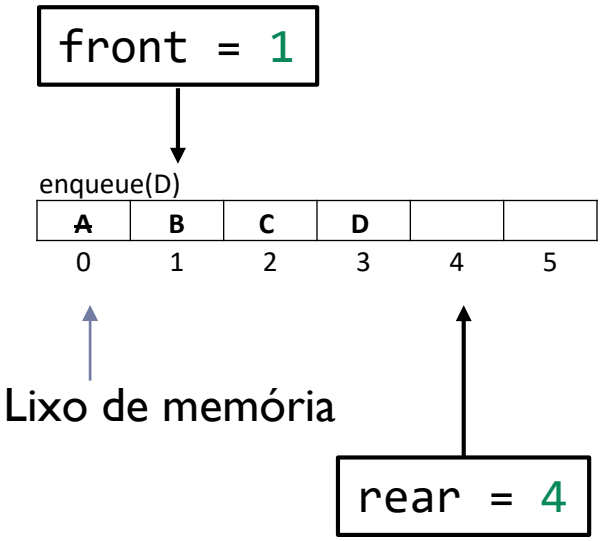
- Como remover?
  - Modificar ponteiro `front`



  - `Problema?`
    - `Com o tempo podemos ficar sem espaço na lista`

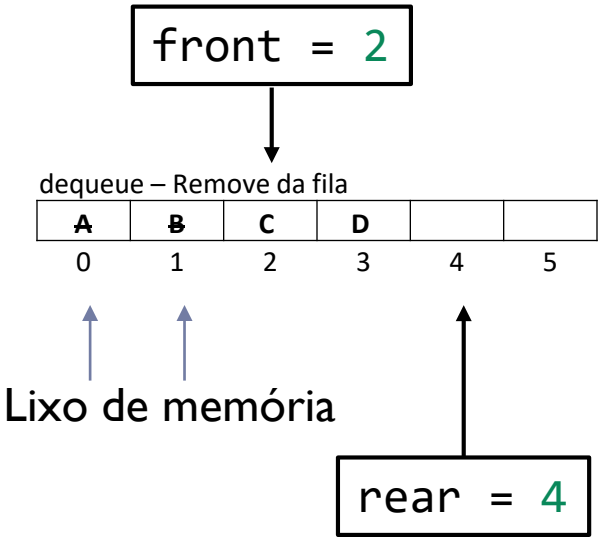
Fila Sequencial Estática

- Problema?
  - Com o tempo podemos ficar sem espaço na lista
  - Insere(D)



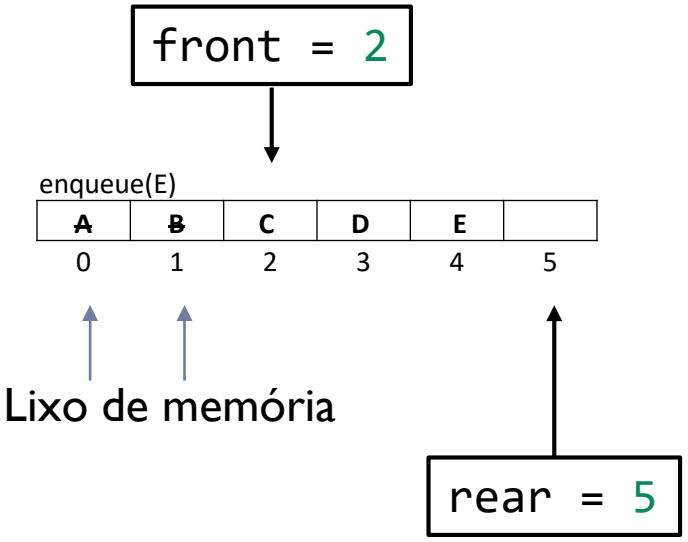
Fila Sequencial Estática

- Problema?
  - Com o tempo podemos ficar sem espaço na lista
  - Insere(D)
  - Remove()



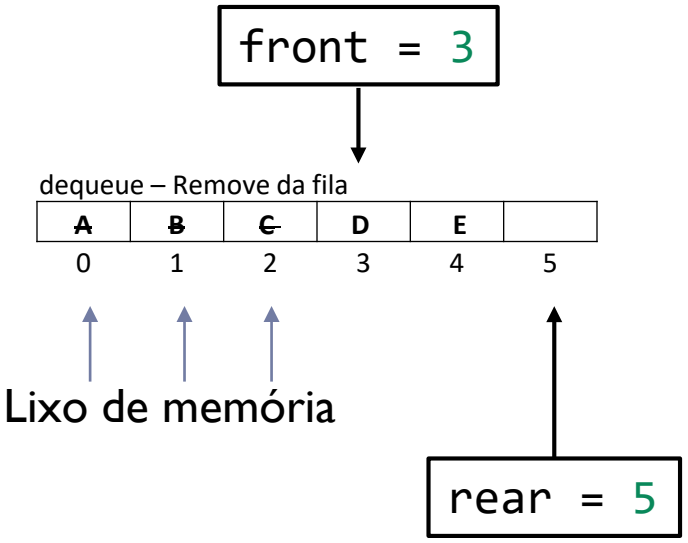
Fila Sequencial Estática

- Problema?
  - Com o tempo podemos ficar sem espaço na lista
  - Insere(D)
  - Remove()
  - Insere(E)



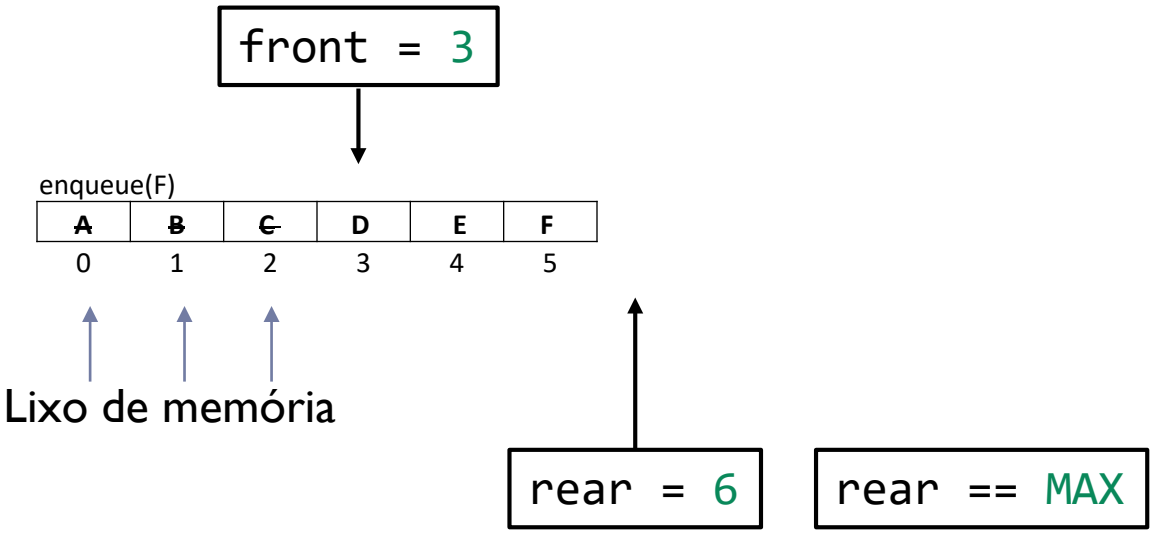
Fila Sequencial Estática

- Problema?
  - Com o tempo podemos ficar sem espaço na lista
  - Insere(D)
  - Remove()
  - Insere(E)
  - Remove()



Fila Sequencial Estática

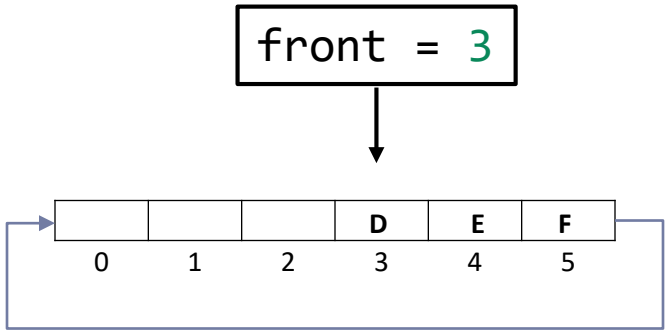
- Problema?
  - Com o tempo podemos ficar sem espaço na lista
  - Insere(D)
  - Remove()
  - Insere(E)
  - Remove()
  - Insere(F)



- A fila já ficou cheia com apenas 3 elementos

Fila Sequencial Estática

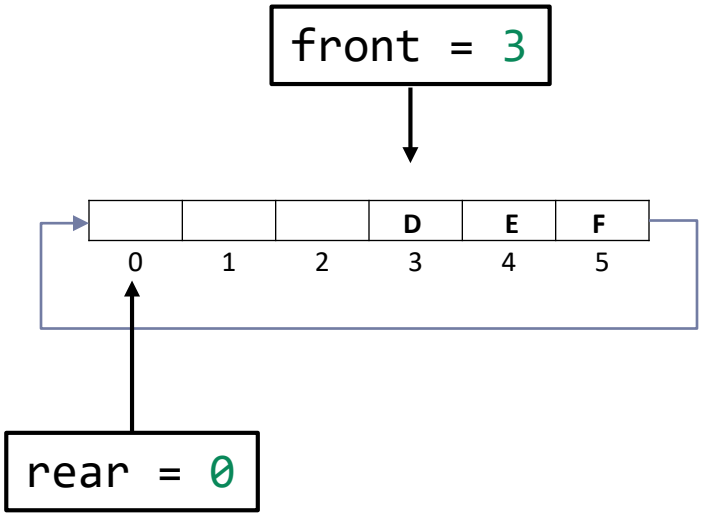
- Problema?
  - Com o tempo podemos ficar sem espaço na lista
  - Usar o vetor de forma circular



  - Quando chegar no último elemento deve-se voltar para o primeiro

Fila Sequencial Estática

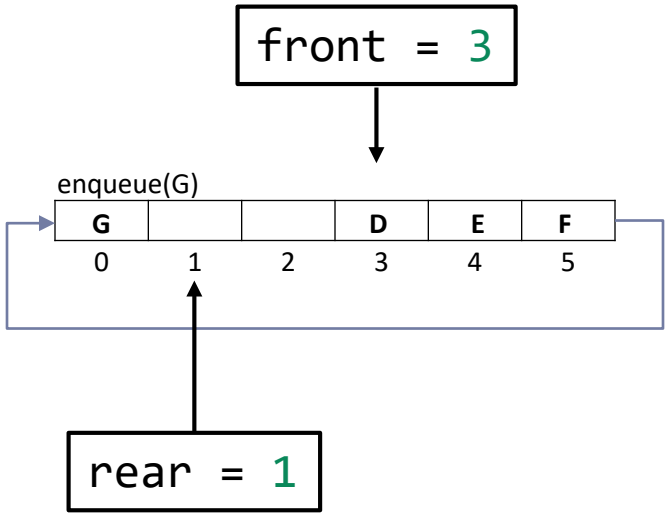
- Problema?
  - Com o tempo podemos ficar sem espaço na lista
  - Usar o vetor de forma circular



  - Quando chegar no ultimo elemento deve-se voltar para o primeiro

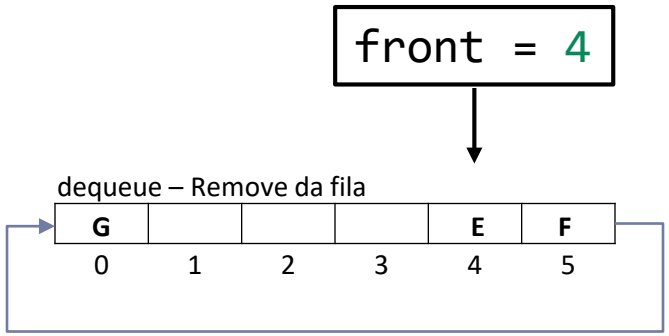
Fila Sequencial Estática

- Vetor Circular
  - Insere(G)



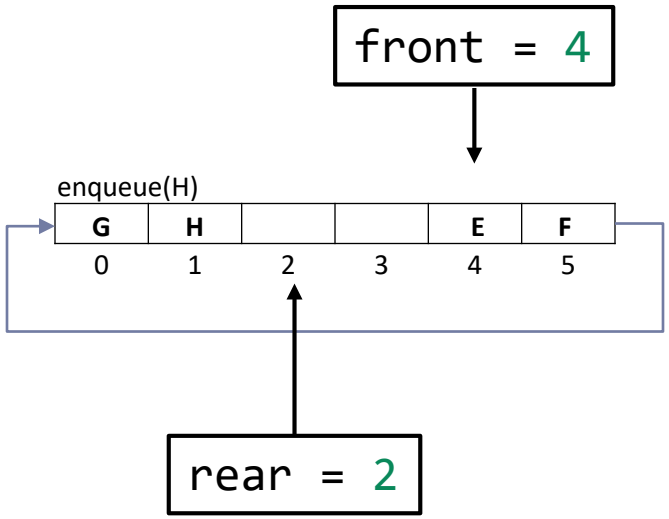
Fila Sequencial Estática

- Vetor Circular
  - Insere(G)
  - Remove()



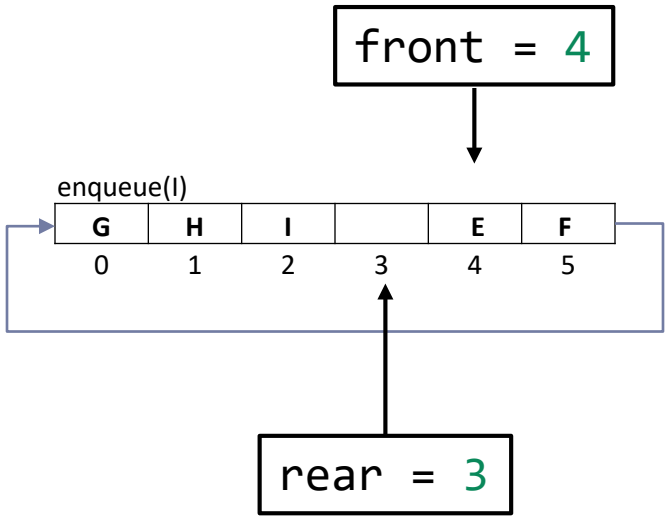
Fila Sequencial Estática

- Vetor Circular
  - Insere(G)
  - Remove
  - Insere(H)



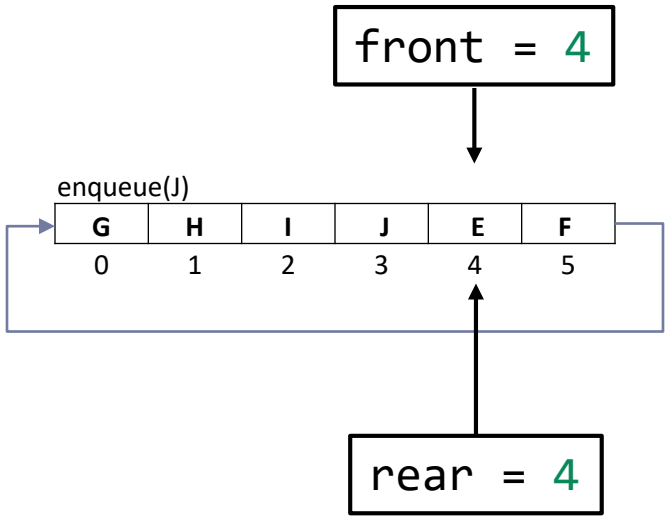
Fila Sequencial Estática

- Vetor Circular
  - Insere(G)
  - Remove
  - Insere(H)
  - Insere(I)



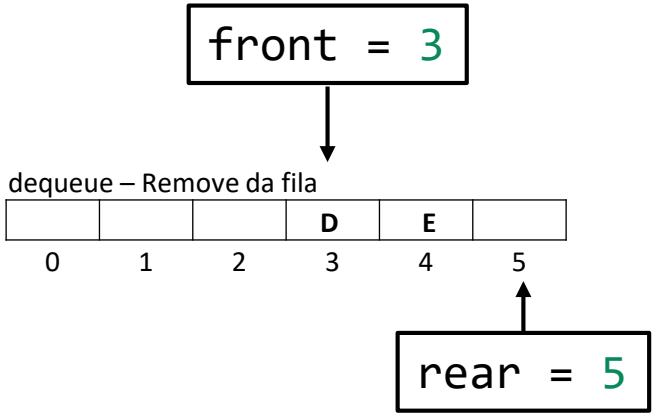
Fila Sequencial Estática

- Vetor Circular
  - Insere(G)
  - Remove
  - Insere(H)
  - Insere(I)
  - Insere(J)



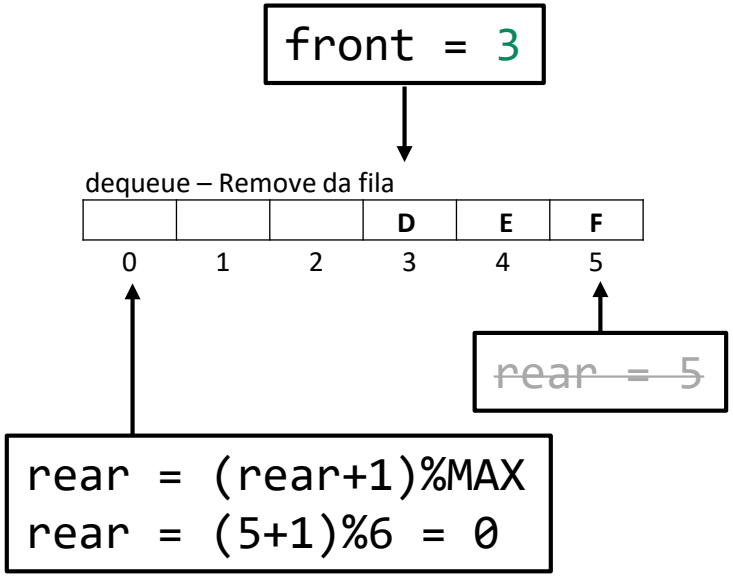
Fila Sequencial Estática

- Como voltar o ‘rear’ para zero?
- Ao inserir um elemento, calcule o resto da divisão de ‘rear’ pelo tamanho do vetor



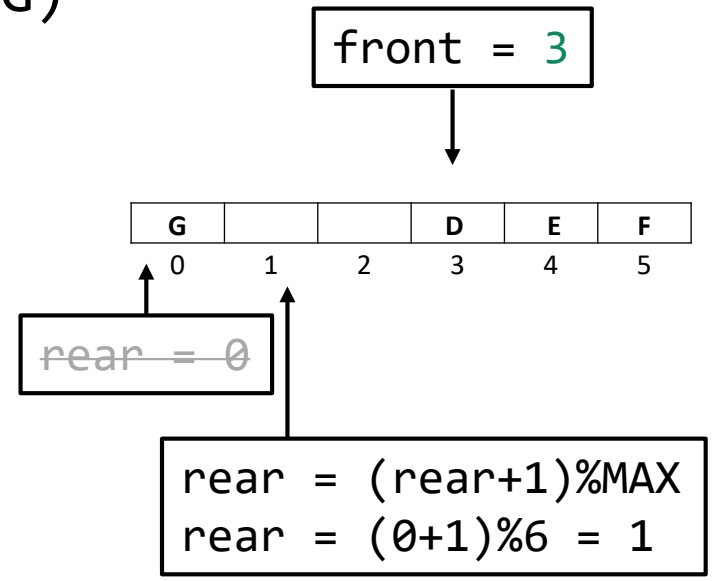
Fila Sequencial Estática

- Como voltar o ‘rear’ para zero?
- Ao inserir um elemento, calcule o resto da divisão de ‘rear’ pelo tamanho do vetor
  - Insere(F)



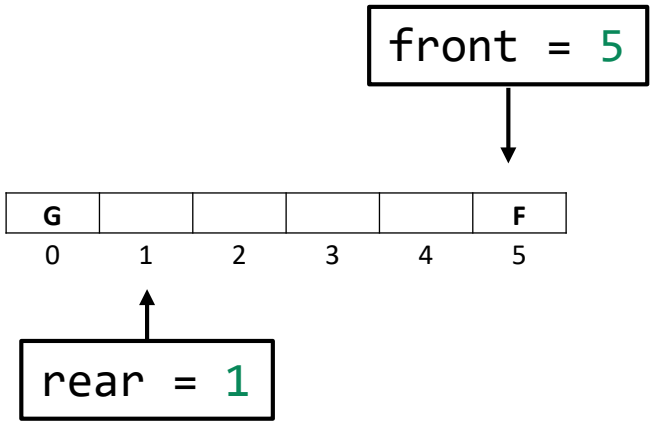
Fila Sequencial Estática

- Como voltar o ‘rear’ para zero?
- Ao inserir um elemento, calcule o resto da divisão de ‘rear’ pelo tamanho do vetor
  - Insere(F)
  - Insere(G)



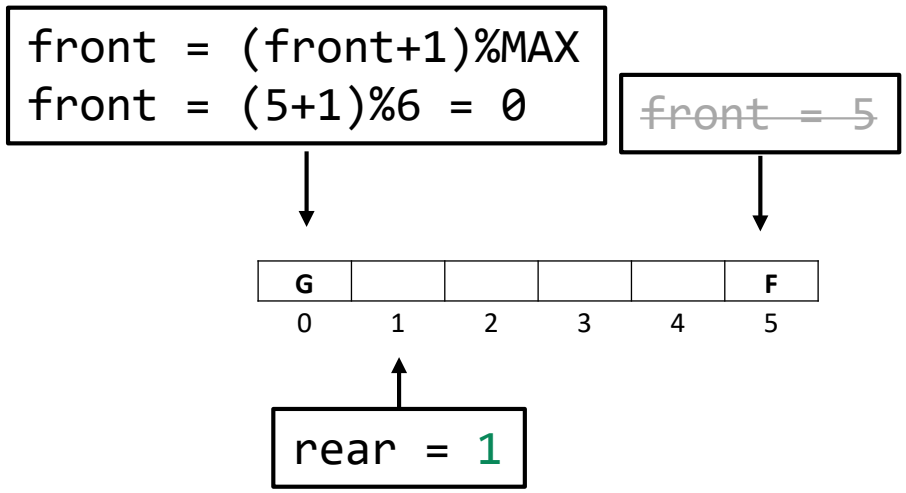
Fila Sequencial Estática

- O mesmo ocorre com o ‘front’



Fila Sequencial Estática

- O mesmo ocorre com o ‘front’
  - Remove()



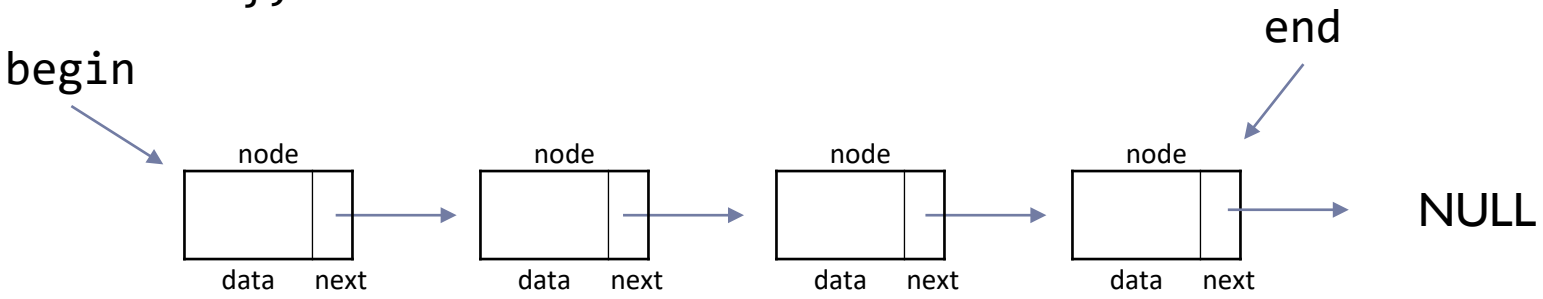
Fila Dinâmica Encadeada

- Estrutura similar a uma lista dinâmica simplesmente encadeada
- Mantém um ponteiro para o último elemento para que a inserção seja rápida

```
typedef struct dqnode DQNode;

struct dqueue {
   DQNode *begin;
   DQNode *end;
   int size;
};

struct dqnode {
   struct aluno data;
   DQNode *next;
};
```

Fila Dinâmica Encadeada

```
typedef struct dqnode DQNode;

struct dqueue {
   DQNode *begin;
   DQNode *end;
   int size;
};

struct dqnode {
   struct aluno data;
   DQNode *next;
};
```



Aplicações

Amazon Simple Queue Service

Fully managed message queues for microservices, distributed systems, and serverless applications

Get started for free

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applic… SQS eliminates the complexity and overhead associated with managing and operating message oriente… middleware, and empowers developers to focus on differentiating work. Using SQS, you can send, store, and r… ve messages between software components at any volume, without losing messages or requiring other services to be available. Get started with SQS in minutes using the AWS console, Command Line Interface or SDK of your choice, and three simple commands.

SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.

https://aws.amazon.com/sqs/