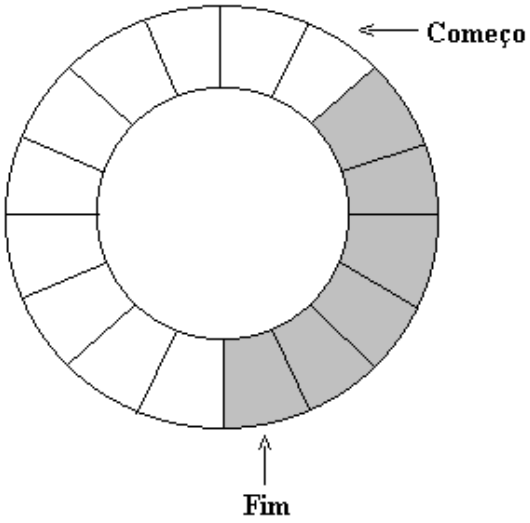
Fila

Fila (Queue) – FIFO

- A Fila ou *Queue* é uma estrutura de dados baseada no princípio FIFO (*First In*, *First Out*), no qual um dado nela inserido em primeiro será removido dela em primeiro:

Fila Circular (Queue-C) – FIFO

- A Fila Circular ou *Queue-C também* é uma estrutura de dados baseada no princípio *FIFO* (*First In*, *First Out*), no qual um dado nela inserido em primeiro será removido dela em primeiro:



Fila Circular (Queue-C) – FIFO

Existem duas funções que se aplicam às filas circulares:

- *enqueueC* : Insere um item no final da fila circular; e

- *dequeueC* : Remove um item do início da fila circular.

Fila Circular (Queue-C) – FIFO

Entrada Fila Circular Saída

E D C B A

Início _____

Fila Circular (Queue-C) – FIFO

Entrada Fila Circular Saída

F E D C B *enqueueC* →

A

Início _____

Fila Circular (Queue-C) – FIFO

Entrada | Fila Circular | Saída

F E D C B

A

Início _____

Fila Circular (Queue-C) – FIFO

Entrada

G F E D C

enqueueC →

Fila Circular

B

A

Início _____

Saída

Fila Circular (Queue-C) – FIFO

Entrada	Fila Circular	Saída
G F E D C		

B

A

Início _____

Fila Circular (Queue-C) – FIFO

Entrada: H G F E D

enqueueC →

Fila Circular:

C

B

A

Início _____

Saída:

Fila Circular (Queue-C) – FIFO

Entrada	Fila Circular	Saída
H G F E D		

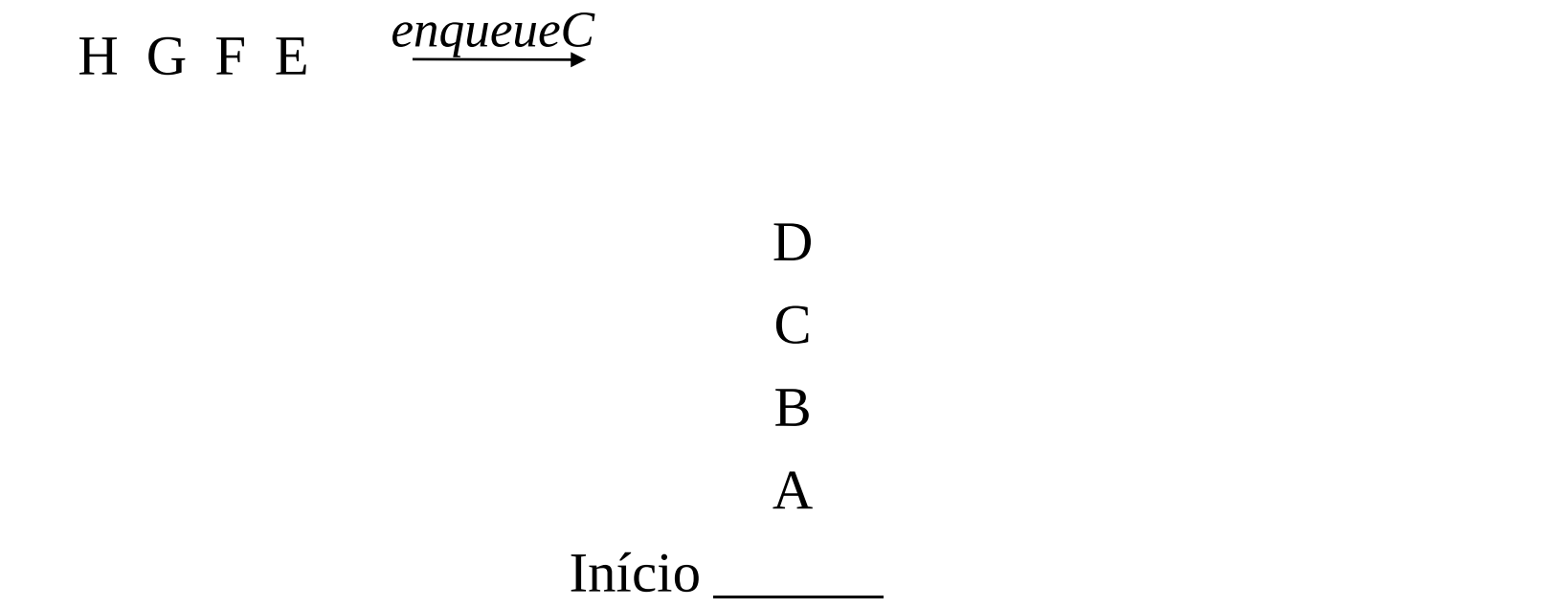
C

B

A

Início _____

Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada

H G F E

Fila Circular

D

C

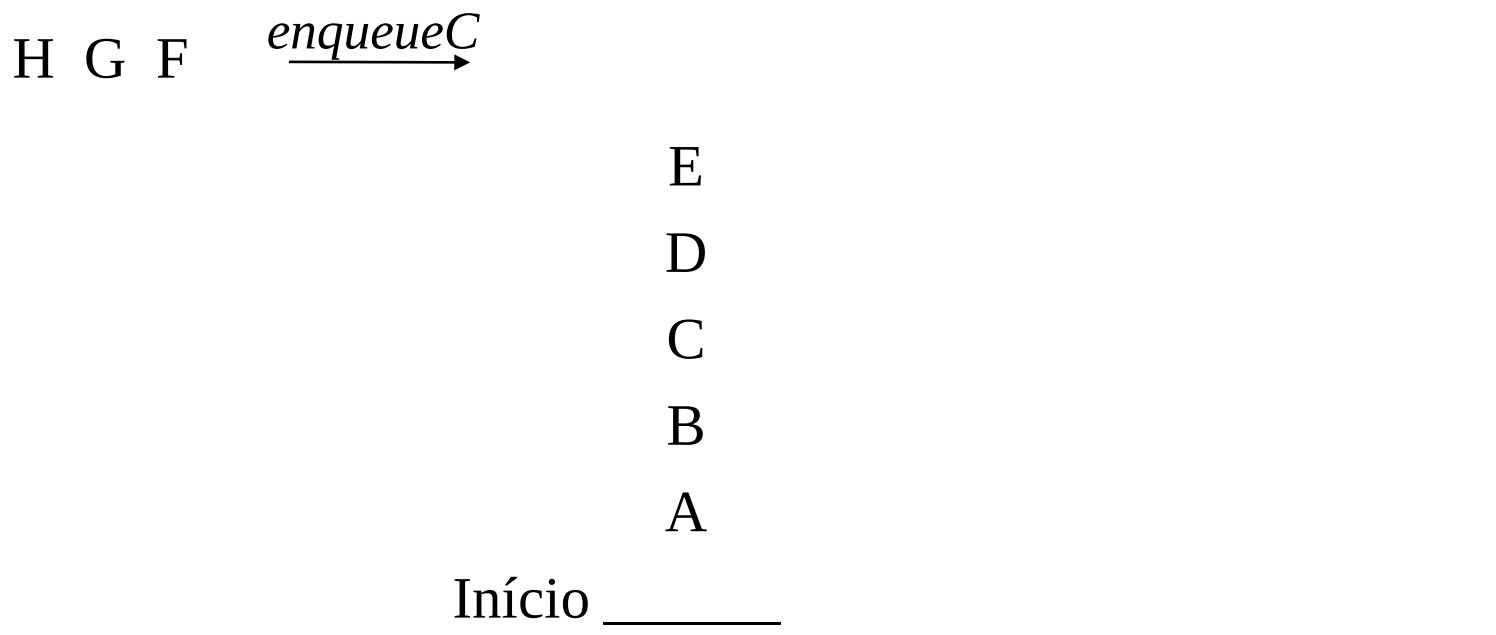
B

A

Início _____

Saída

Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada | Fila Circular | Saída

H G F

E

D

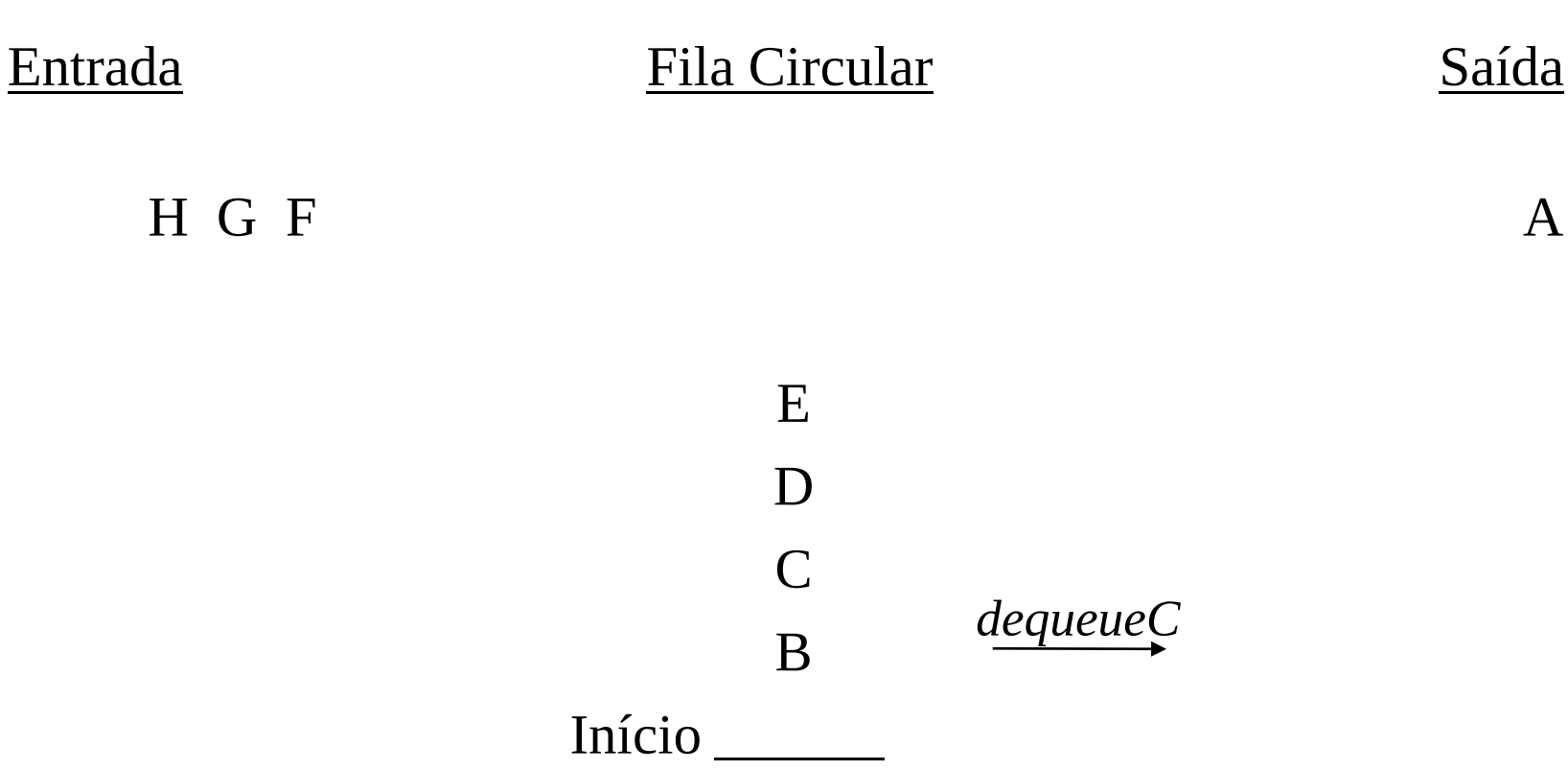
C

B

A

Início _____

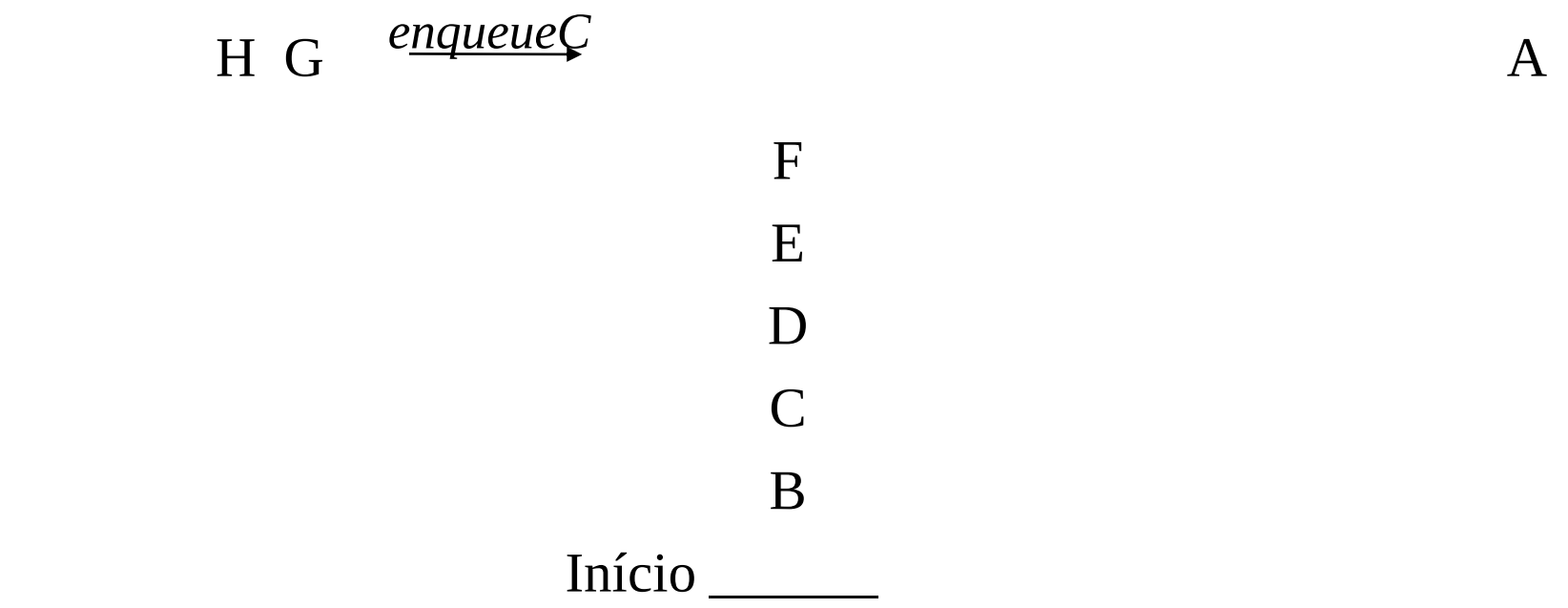
Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada	Fila Circular	Saída
H G F		A
	E	
	D	
	C	
	B	
	Início _____	

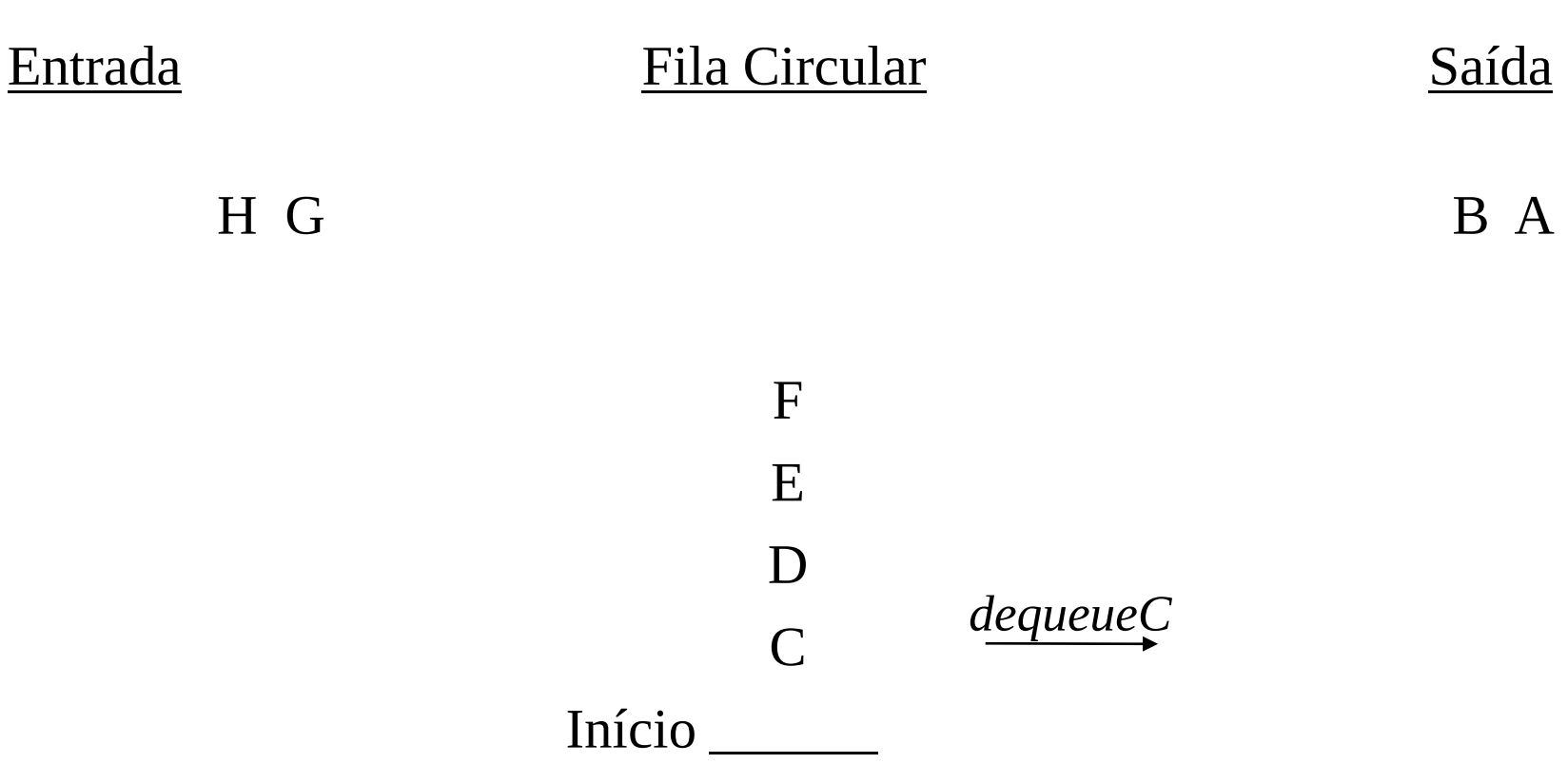
Fila Circular (Queue-C) – FIFO



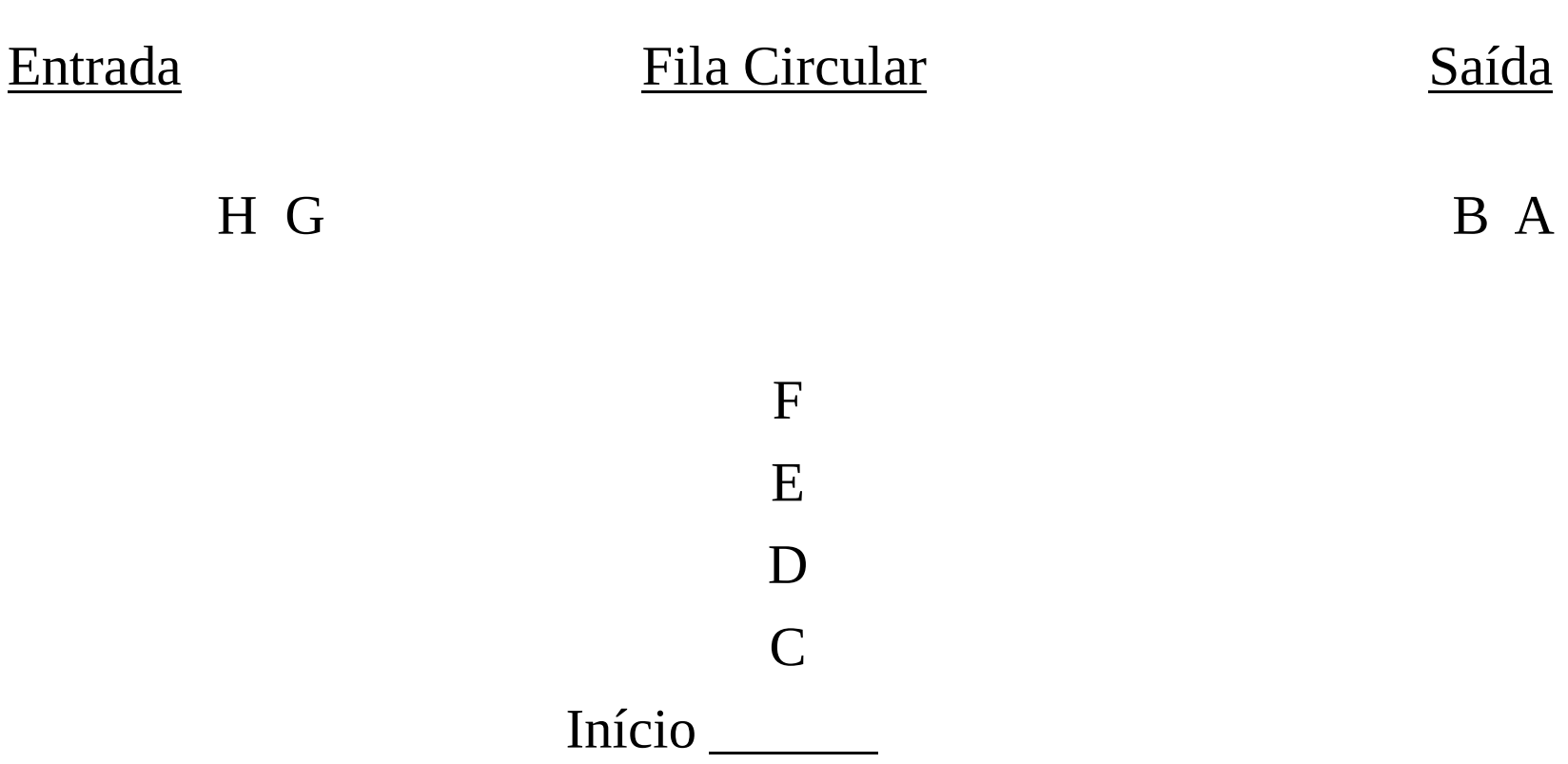
Fila Circular (Queue-C) – FIFO

Entrada	Fila Circular	Saída
H G		A
	F	
	E	
	D	
	C	
	B	
	Início _____	

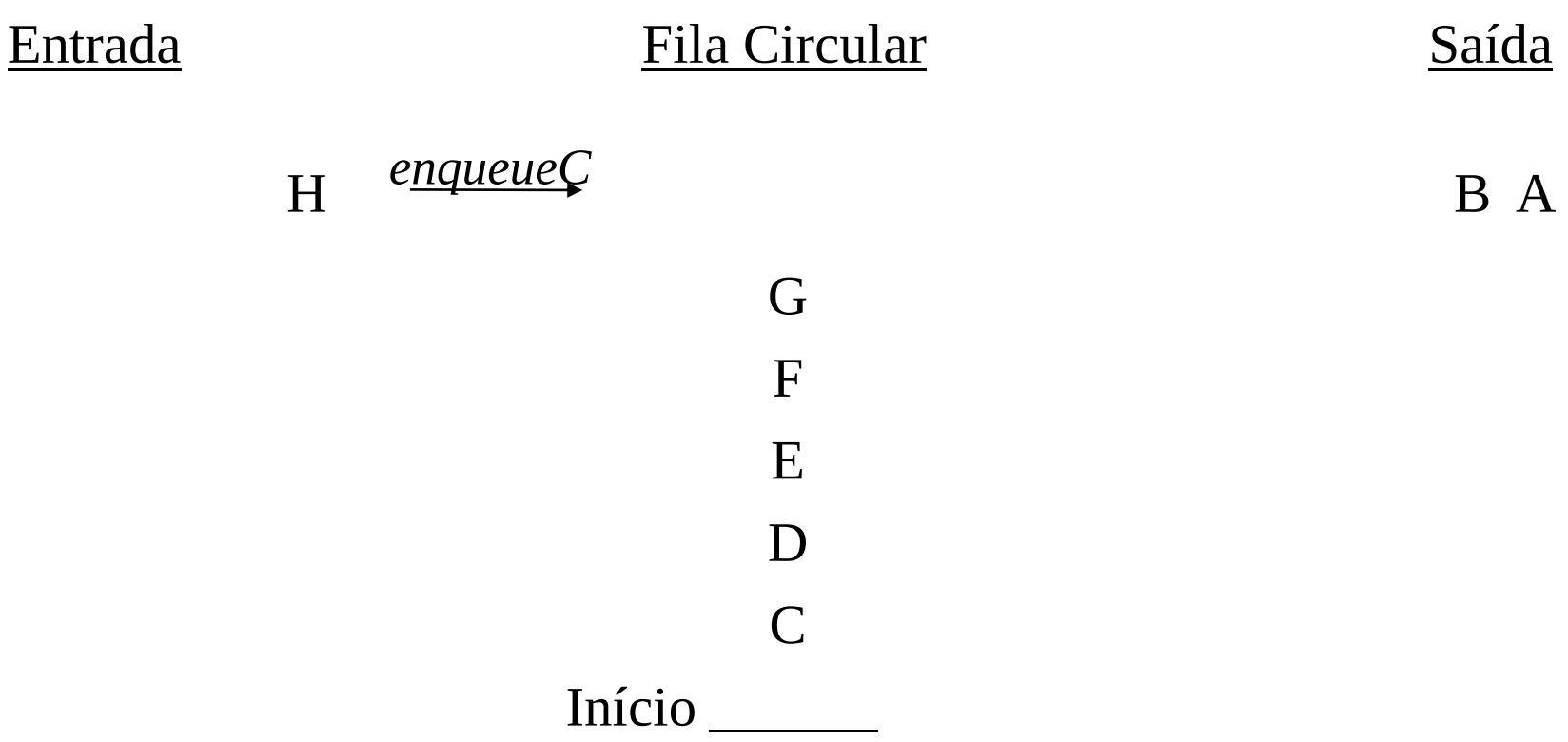
Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO



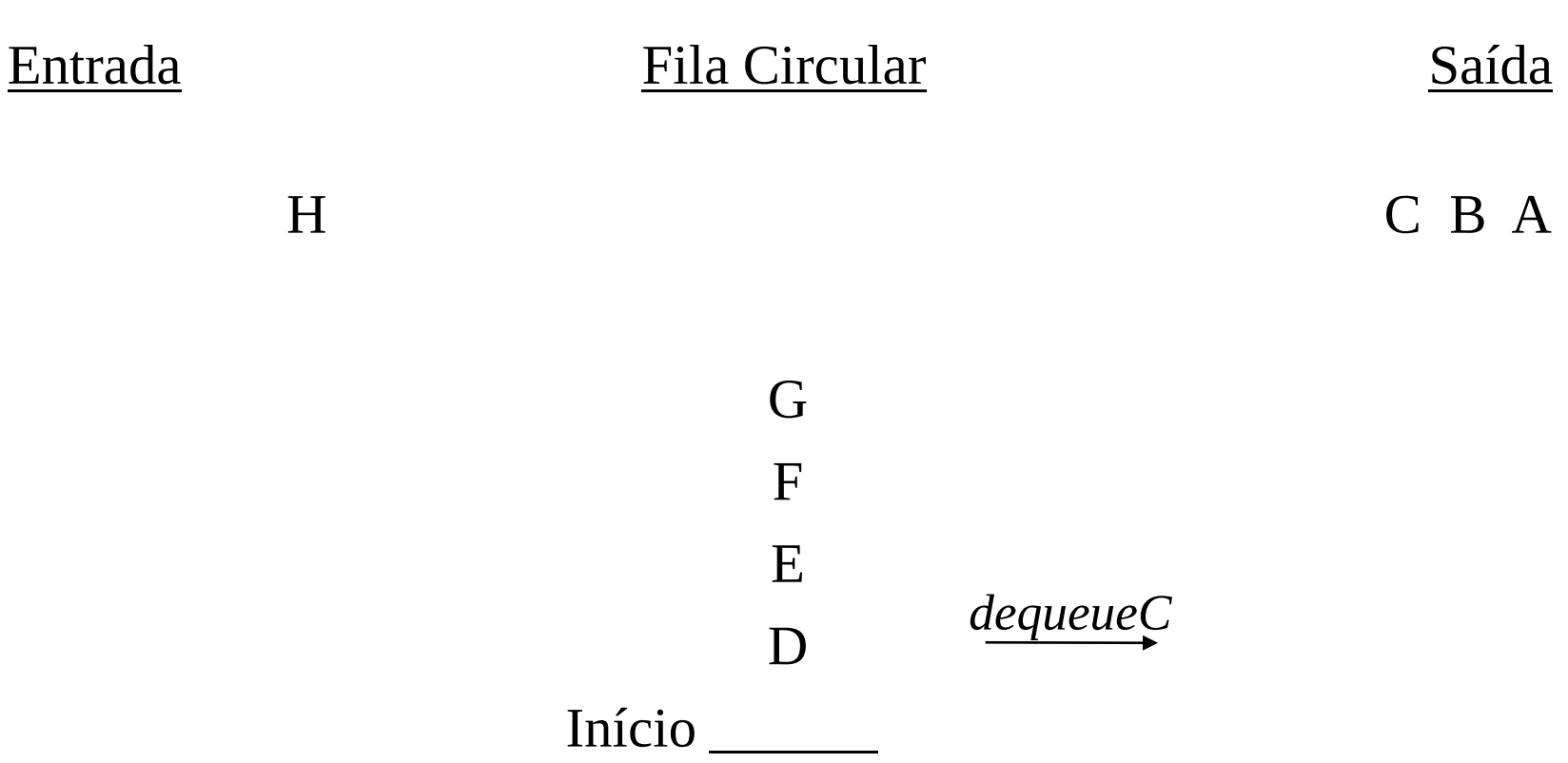
Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada	Fila Circular	Saída
H	G F E D C Início _____	B A

Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada

H

Fila Circular

G

F

E

D

Início _____

Saída

C B A

Fila Circular (Queue-C) – FIFO

Entrada

Fila Circular

Saída

*enqueue*C

C B A

H

G

F

E

D

Início _____

Fila Circular (Queue-C) – FIFO

Entrada

Fila Circular

Saída

C B A

H

G

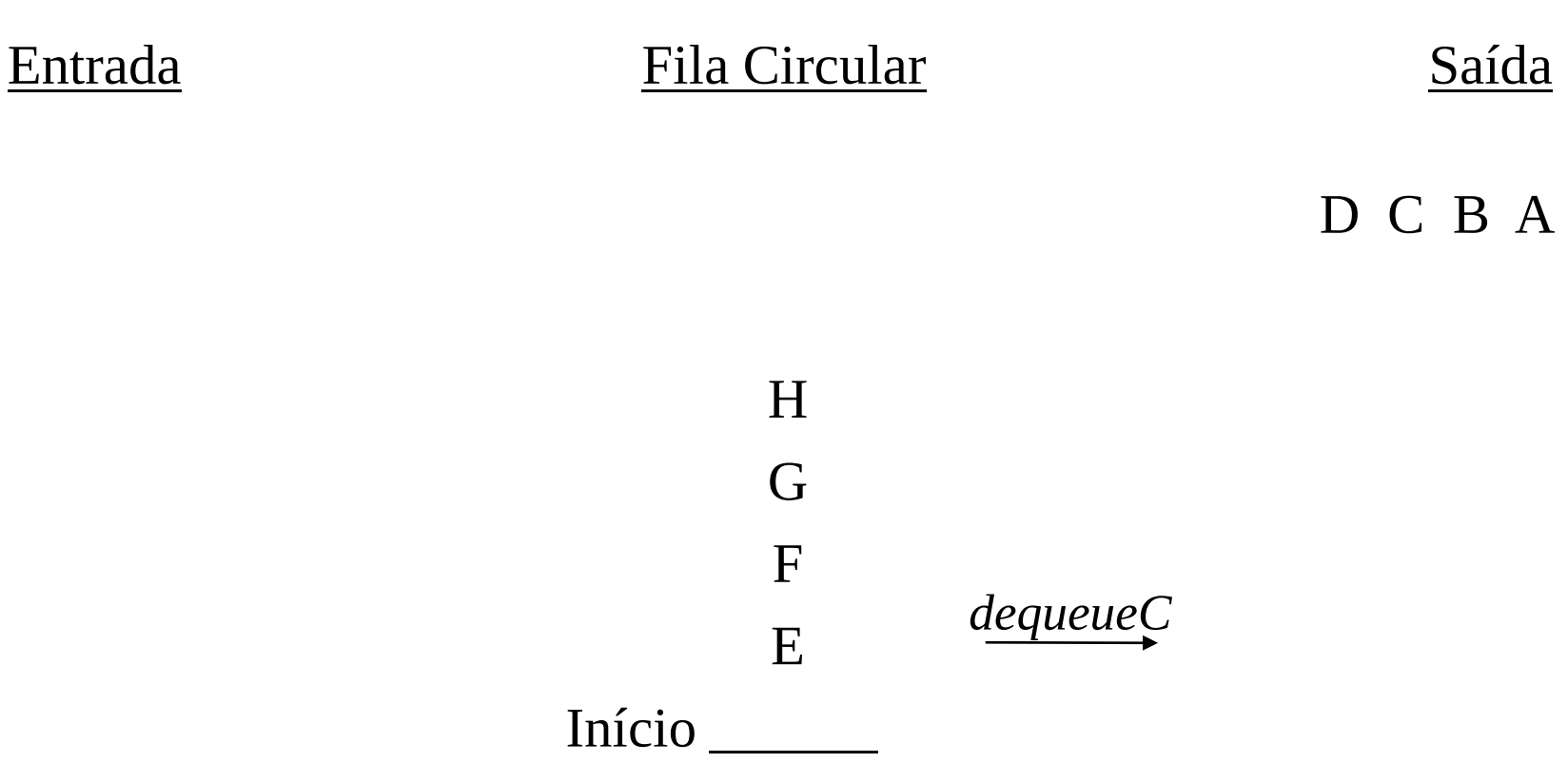
F

E

D

Início _____

Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada

Fila Circular

Saída

D C B A

H

G

F

E

Início _____

Fila Circular (Queue-C) – FIFO

Entrada | Fila Circular | Saída

E D C B A

H
G
F

dequeueC →

Início _____

Fila Circular (Queue-C) – FIFO

Entrada

Fila Circular

Saída

E D C B A

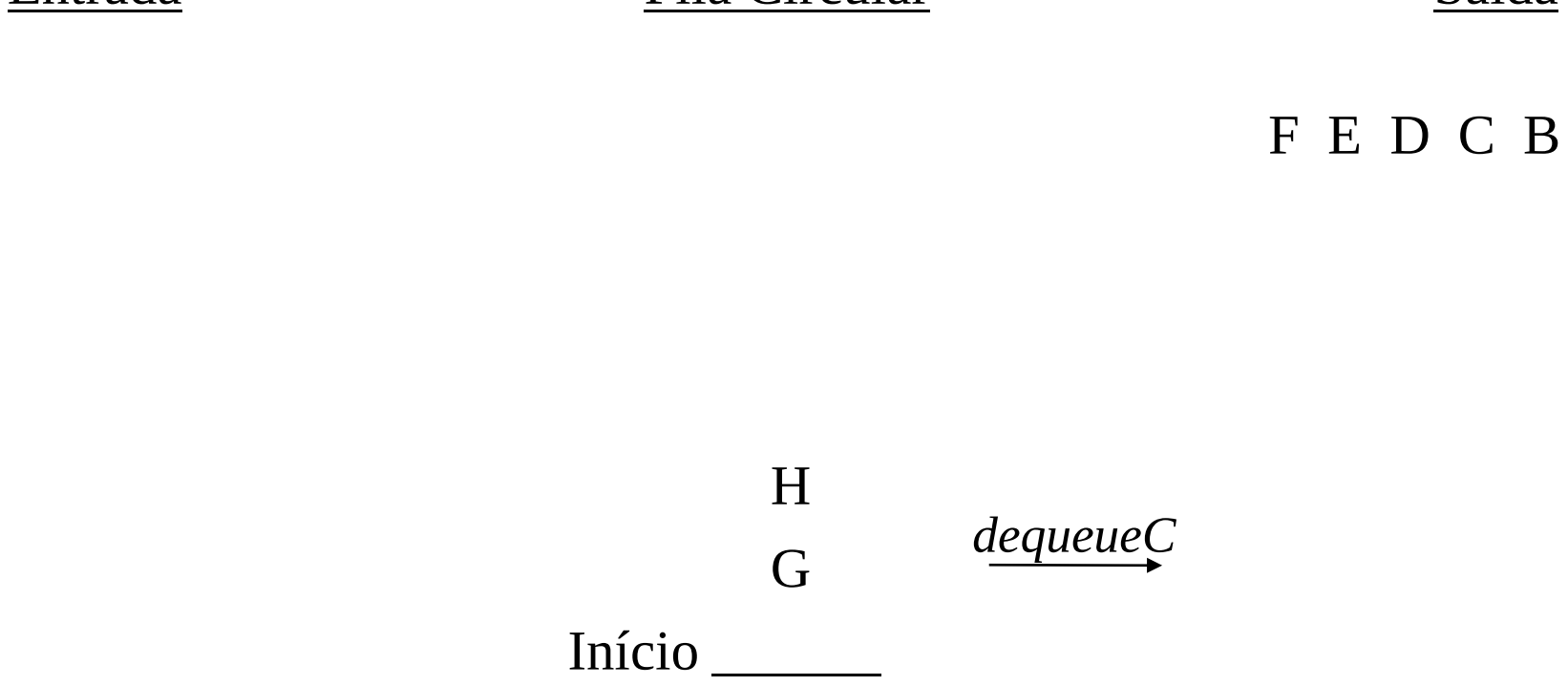
H

G

F

Início _____

Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada | Fila Circular | Saída

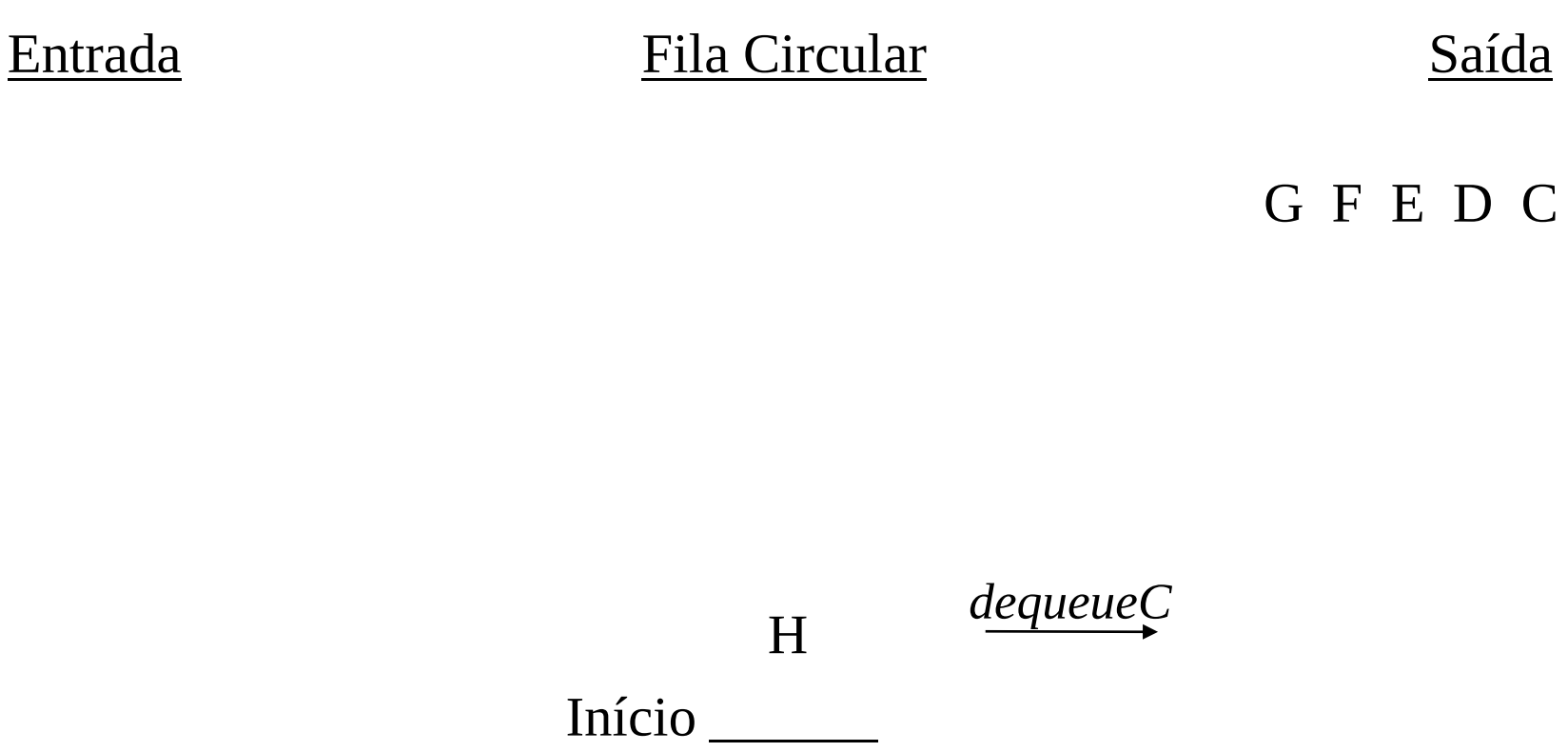
F E D C B

H

G

Início _____

Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Entrada | Fila Circular | Saída

G F E D C

H

Início _____

Fila Circular (Queue-C) – FIFO

Entrada Fila Circular Saída

H G F E D

dequeueC

Início _____

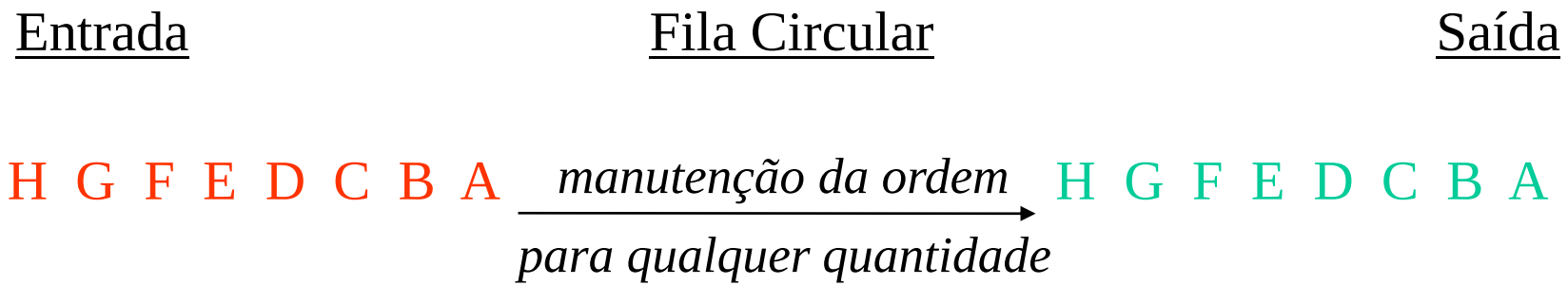
Fila Circular (Queue-C) – FIFO

Entrada | Fila Circular | Saída

H G F E D

Início _____

Fila Circular (Queue-C) – FIFO



Fila Circular (Queue-C) – FIFO

Exemplos de utilização de filas circulares:

- Execução dos processos de um sistema operacional;

- *Buffers* de informação (impressoras, HDs etc.); e

- Pacotes de mensagens em um roteador.

Fila Circular (Queue-C) – FIFO

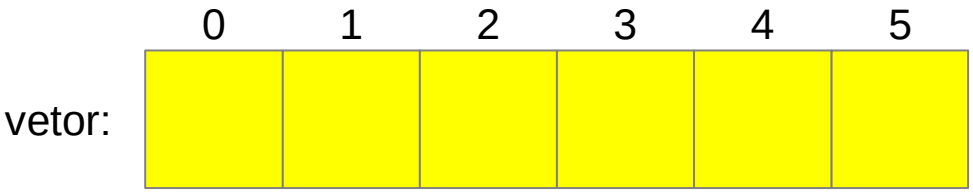
Outras funções que se aplicam às filas circulares:

- *size* : Informa o tamanho da referida fila circular;

- *front*: Informa o elemento no início da fila circular, sem retirá-lo; e

- *isOver*: Informa se a fila circular está cheia.

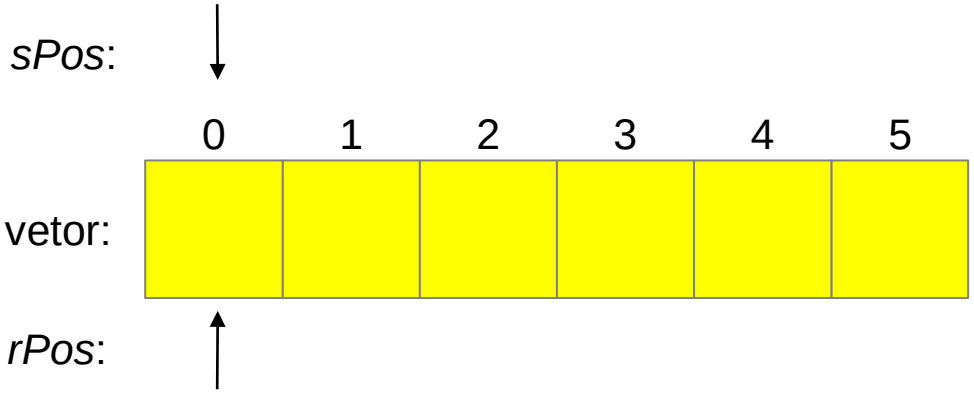
Fila Circular (Queue-C) – FIFO



int vetor[ ] = new vetor[6];

Vetor com o tamanho máximo necessário

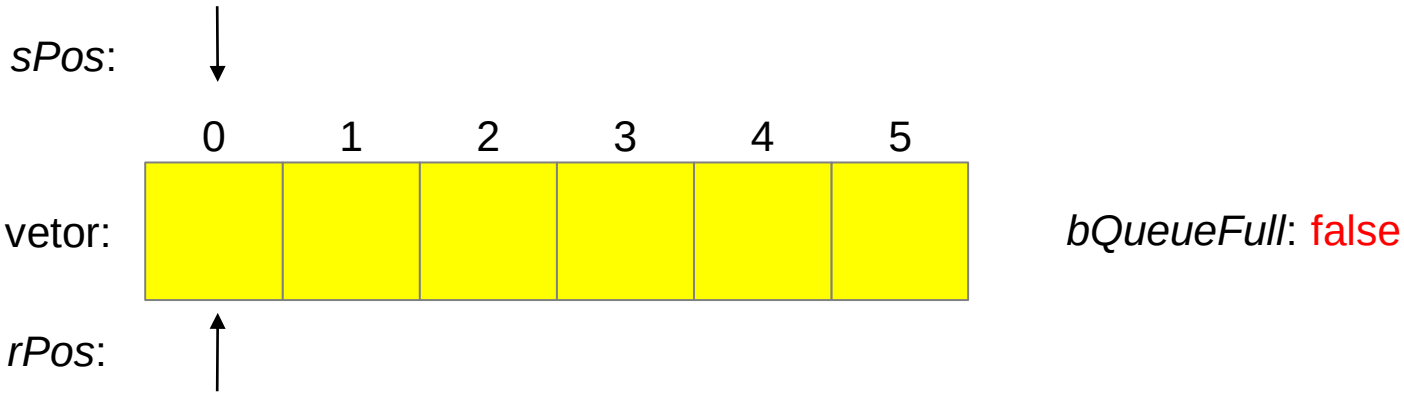
Fila Circular (Queue-C) – FIFO




```
int sPos, rPos;
:
:
sPos = rPos = 0;
```

Inicializa indicadores de posição

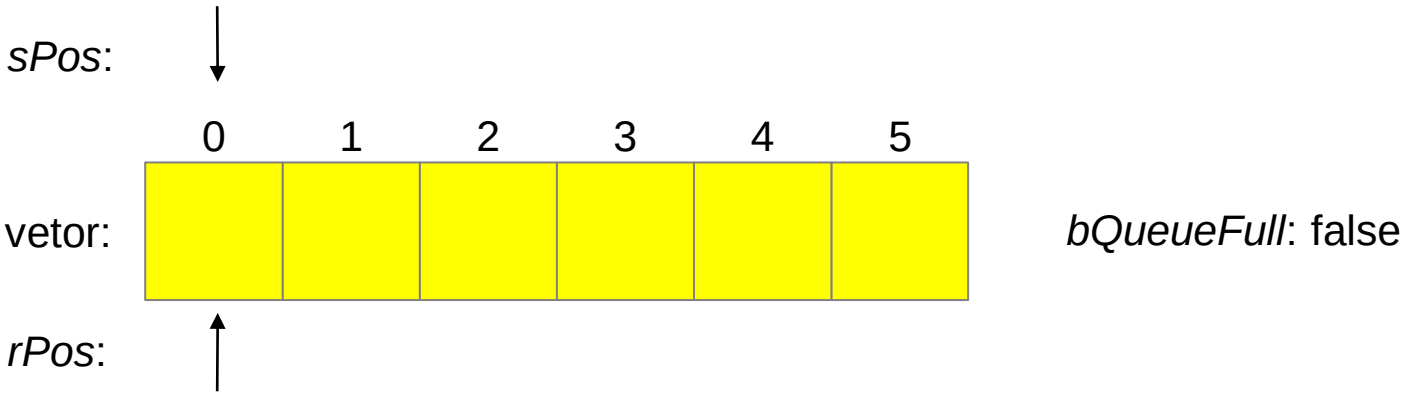
Fila Circular (Queue-C) – FIFO




```
boolean bQueueFull;
:
:
bQueueFull = false;
```

Inicializa indicador de fila circular cheia

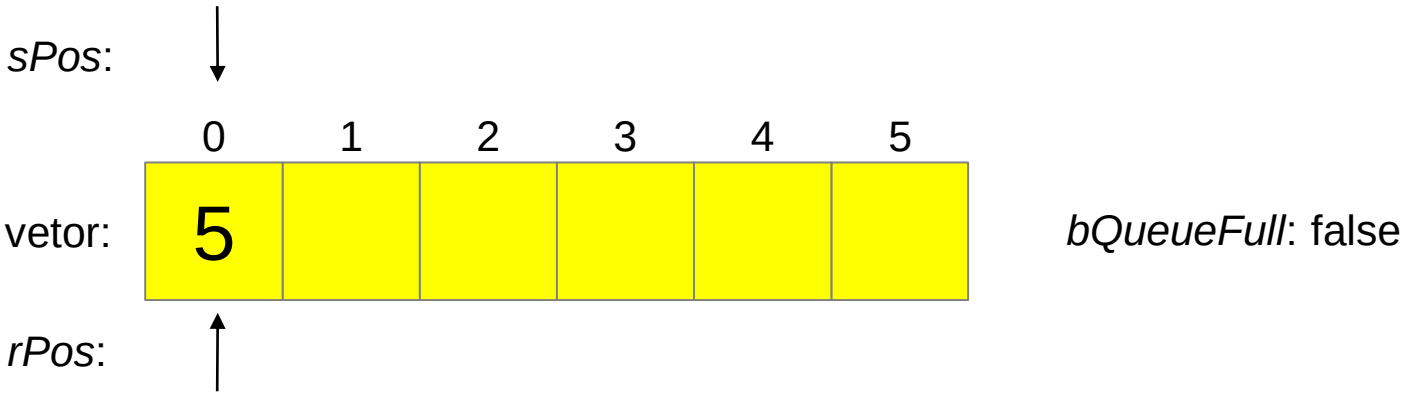
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(5);
```

Verifica se está cheia

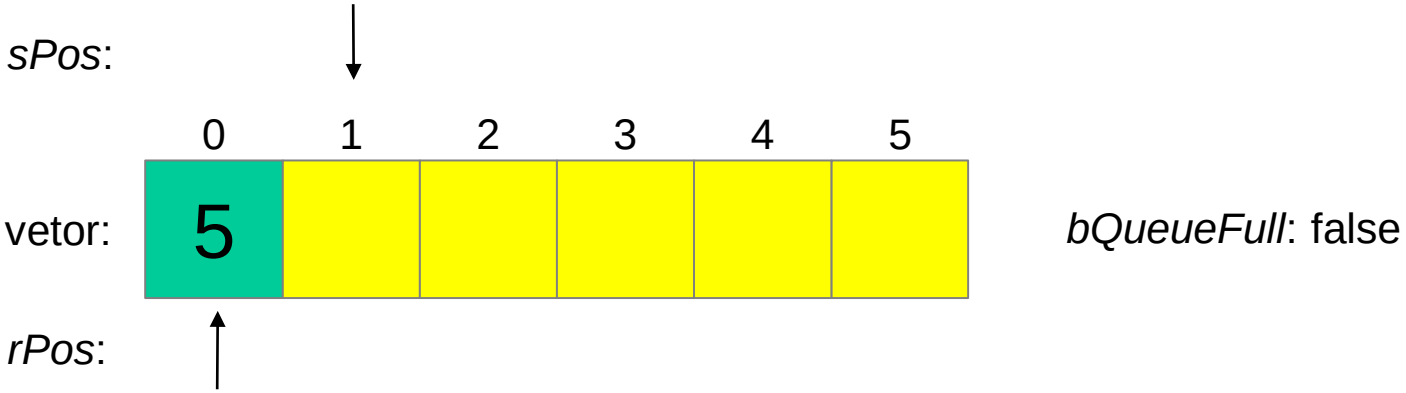
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(5);
```

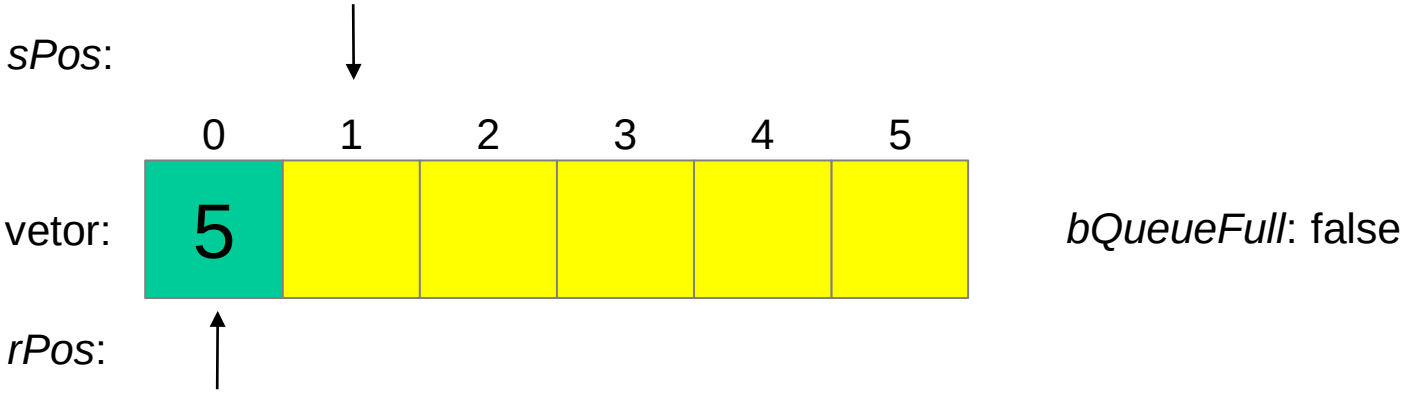
Não está cheia! Coloca o elemento 5

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

Fila Circular (Queue-C) – FIFO

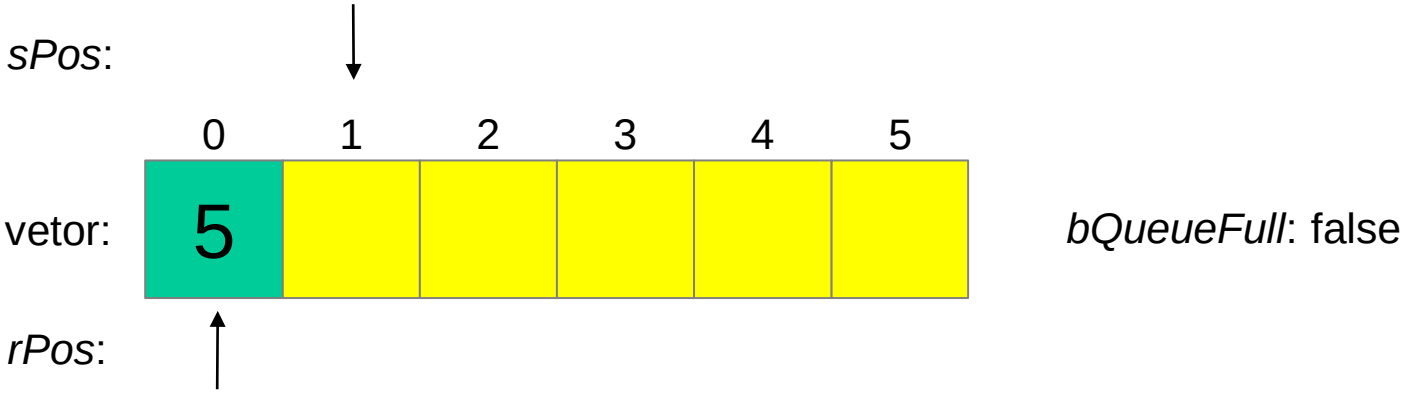



```
int frente;
:
:
frente = front( );

//frente: 5
```

Consulta a frente

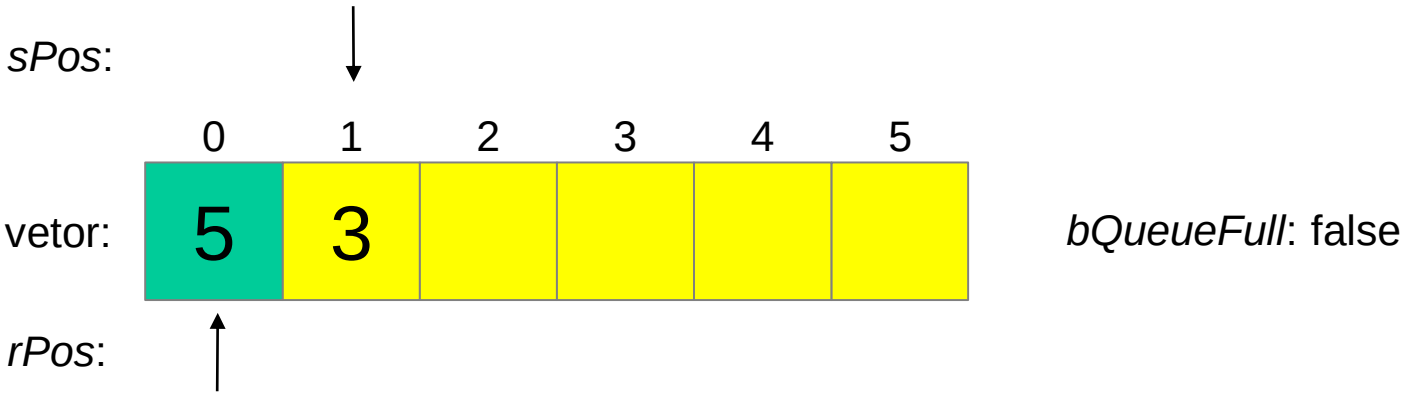
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println(“Fila Circular Cheia”);
else
    enqueueC(3);
```

Verifica se está cheia

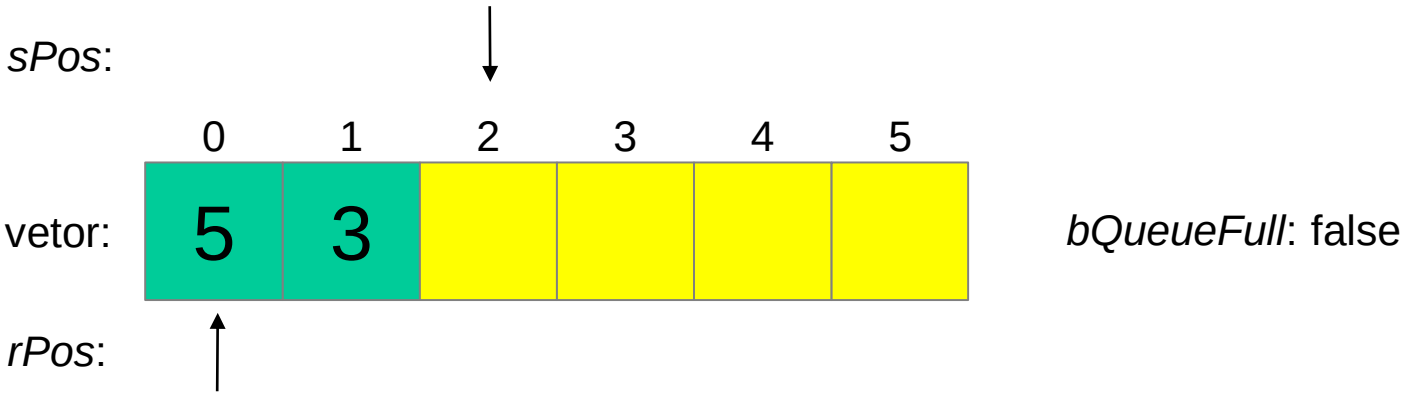
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(3);
```

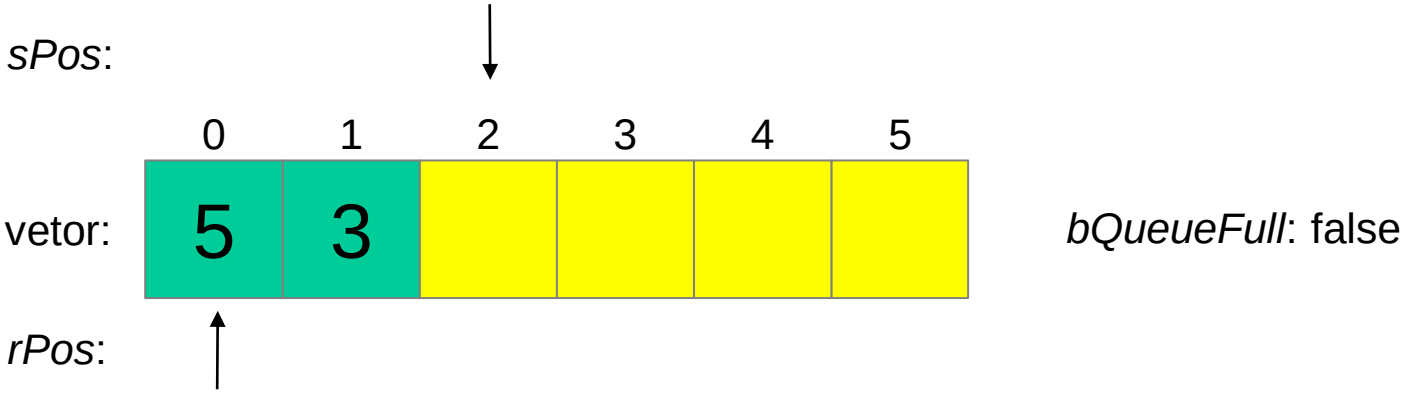
Não está cheia! Coloca o elemento 3

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

Fila Circular (Queue-C) – FIFO

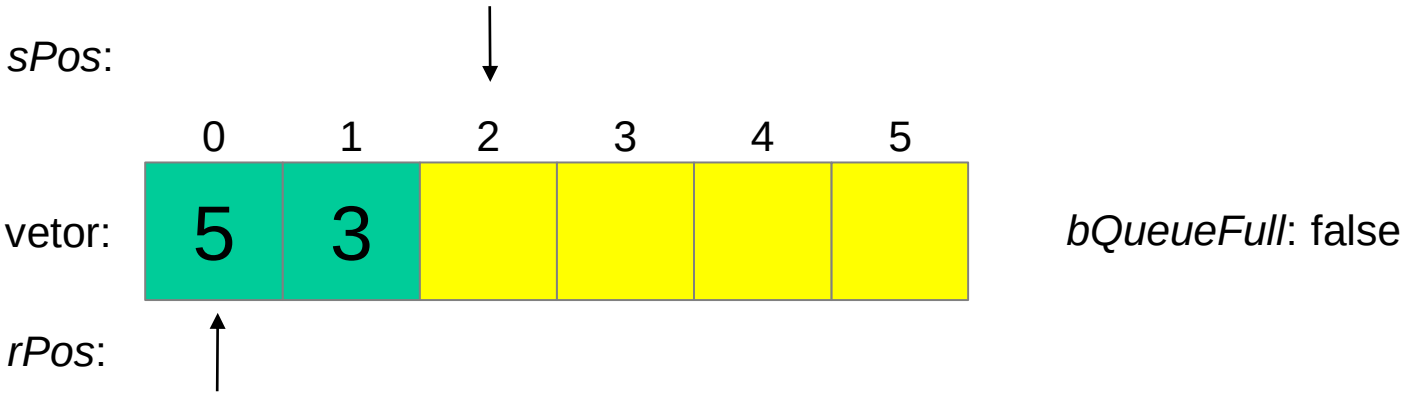



```
int frente;
:
:
frente = front( );

//frente: 5
```

Consulta a frente

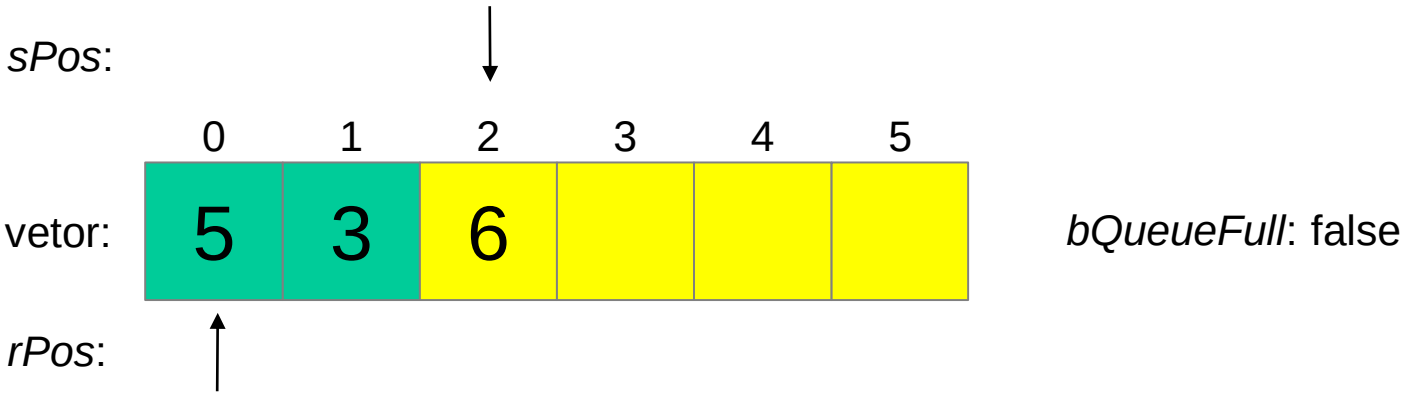
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(6);
```

Verifica se está cheia

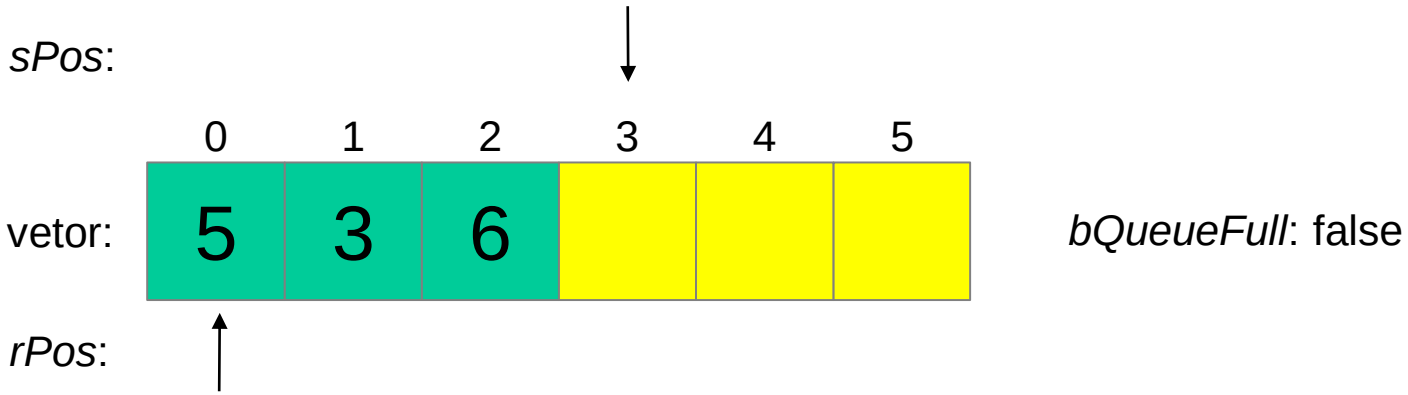
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println(“Fila Circular Cheia”);
else
    enqueueC(6);
```

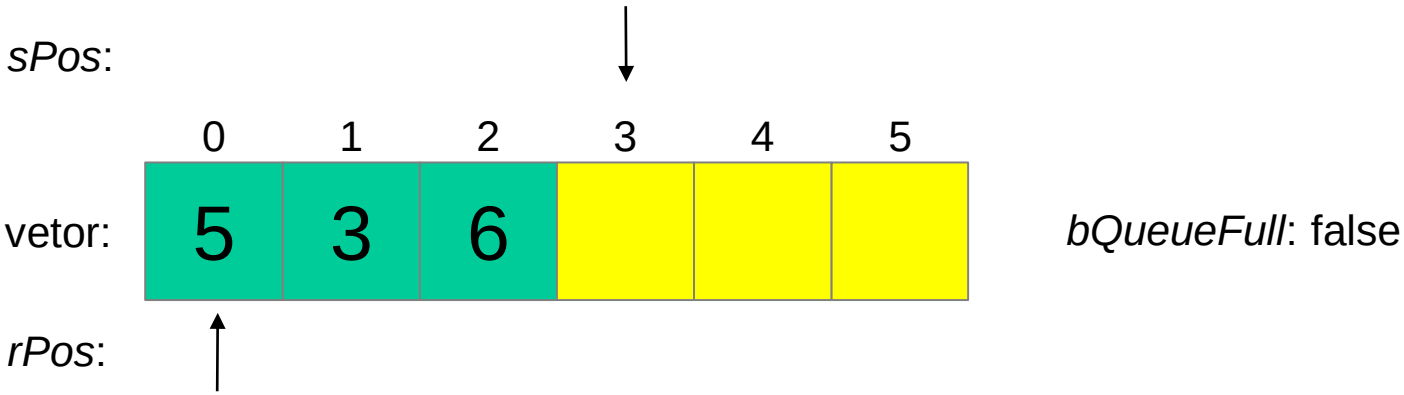
Não está cheia! Coloca o elemento 6

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

Fila Circular (Queue-C) – FIFO

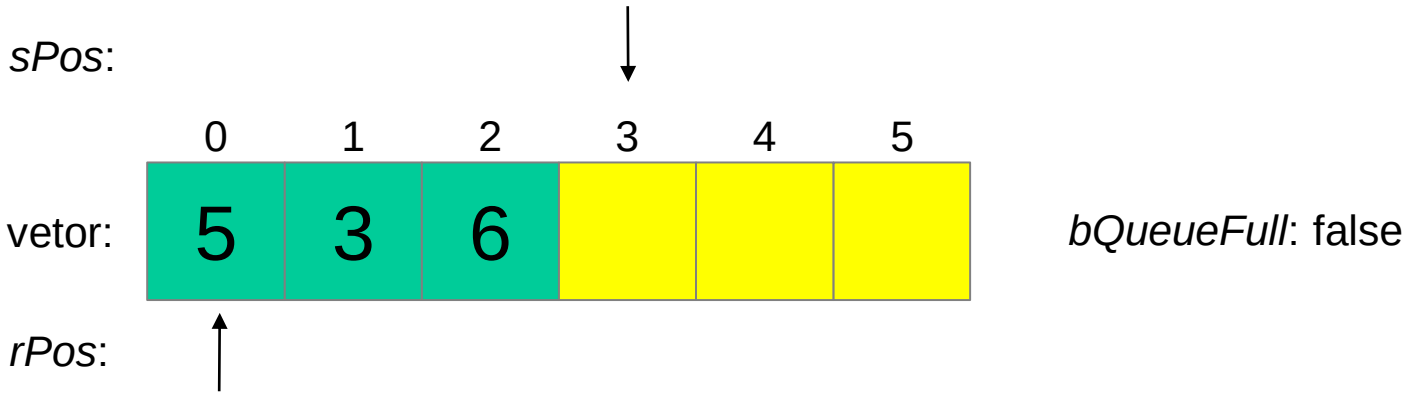



```
int frente;
:
:
frente = front( );

//frente: 5
```

Consulta a frente

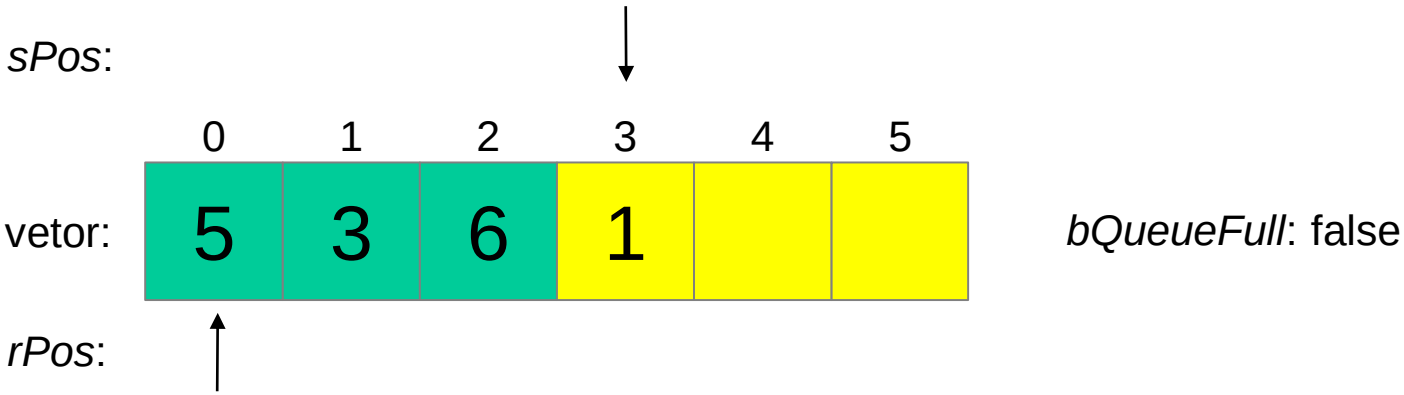
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(1);
```

Verifica se está cheia

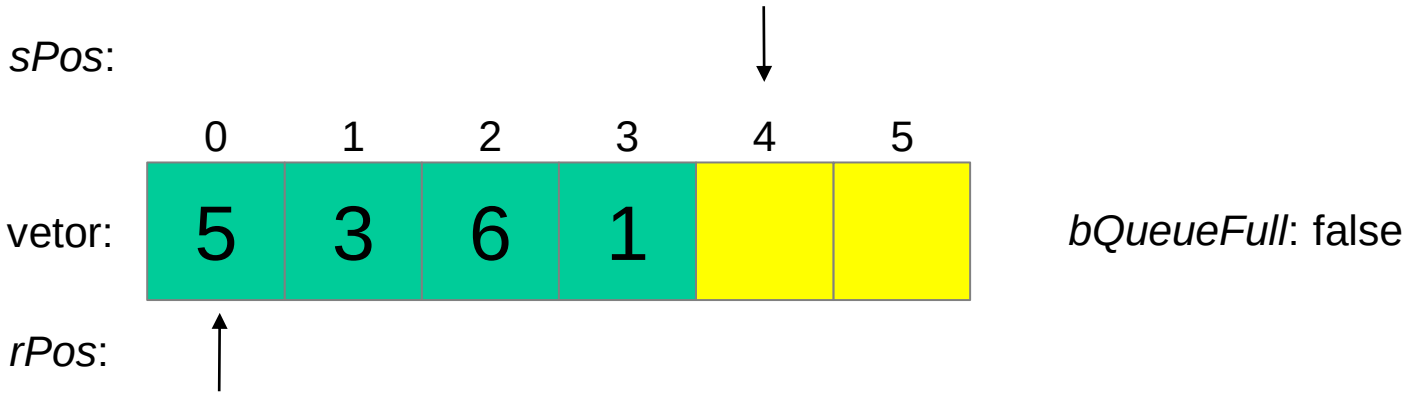
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(1);
```

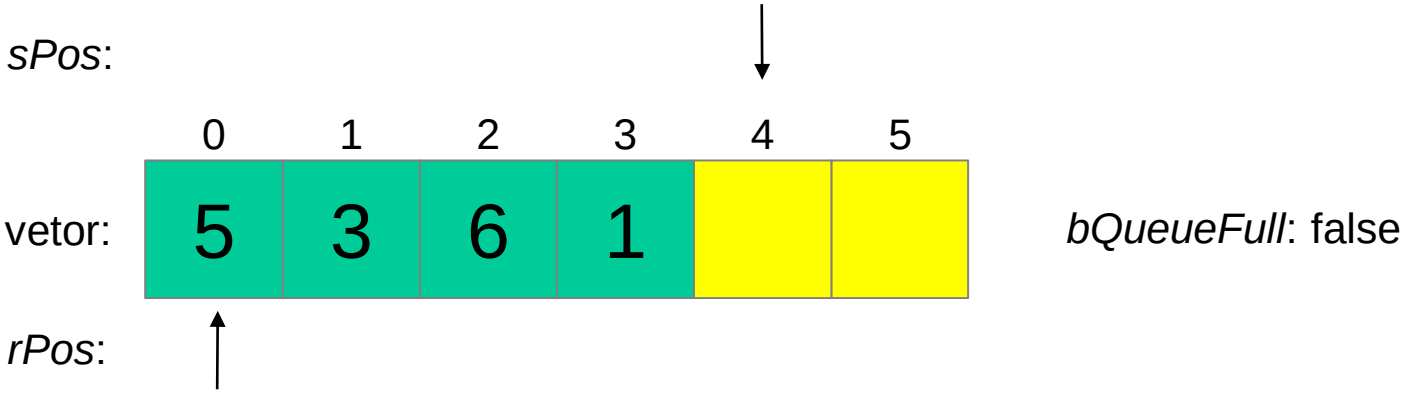
Não está cheia! Coloca o elemento 1

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

Fila Circular (Queue-C) – FIFO

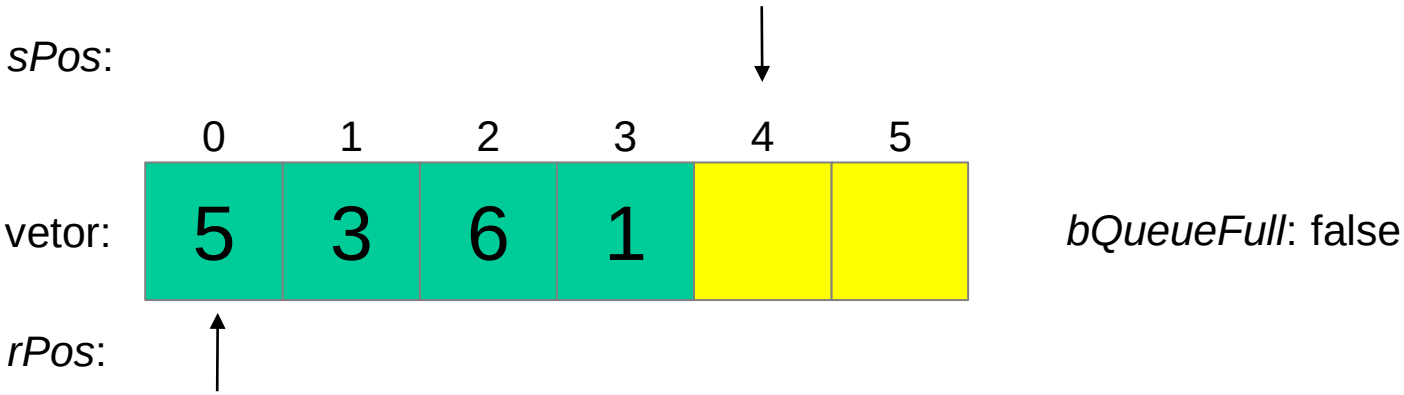



```
int frente;
:
:
frente = front( );

//frente: 5
```

Consulta a frente

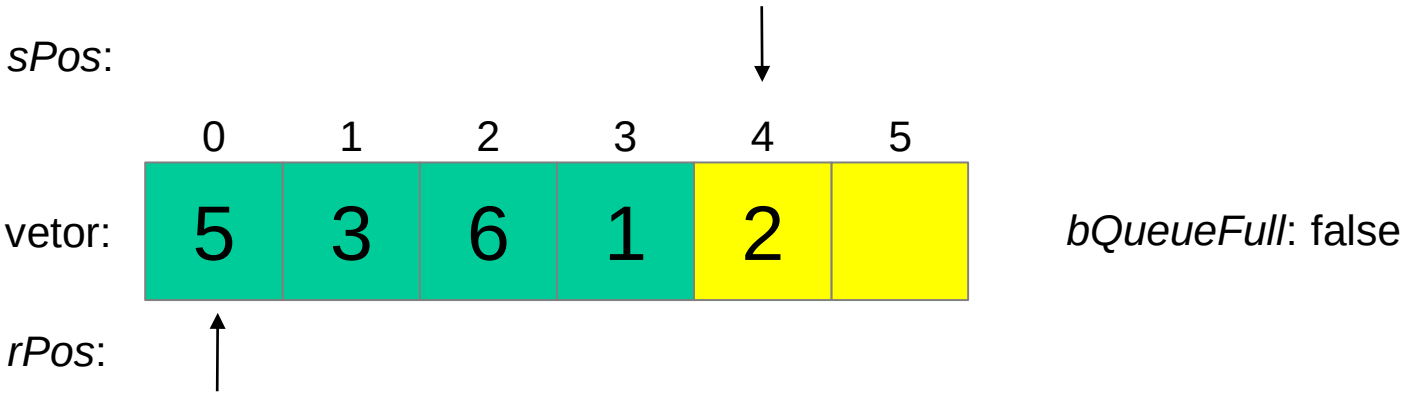
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(2);
```

Verifica se está cheia

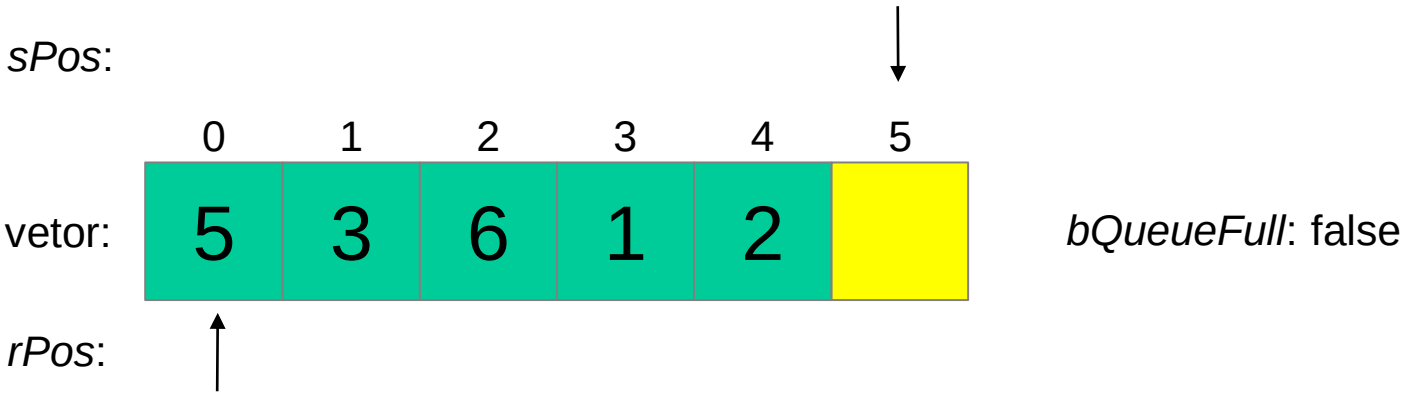
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(2);
```

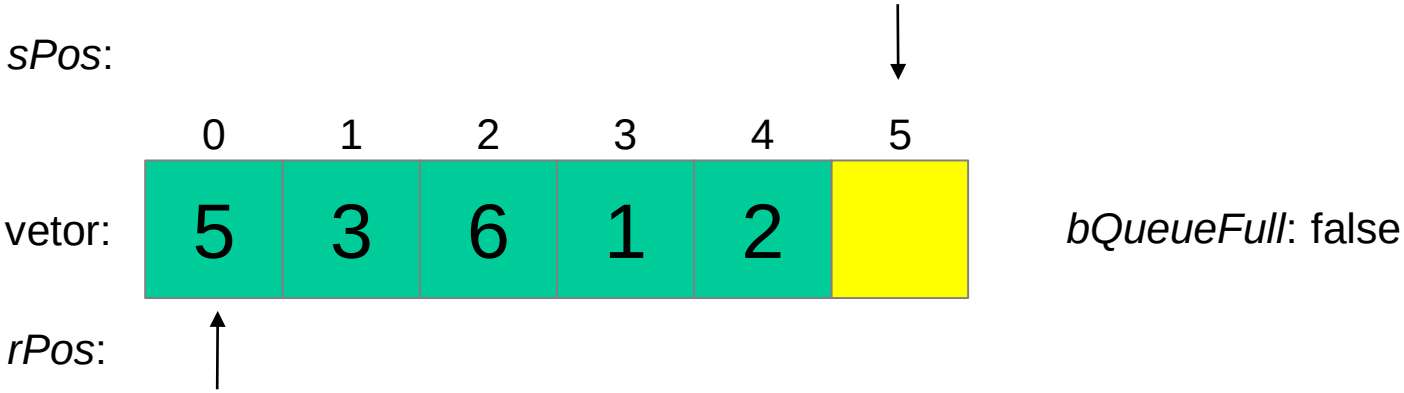
Não está cheia! Coloca o elemento 2

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

Fila Circular (Queue-C) – FIFO

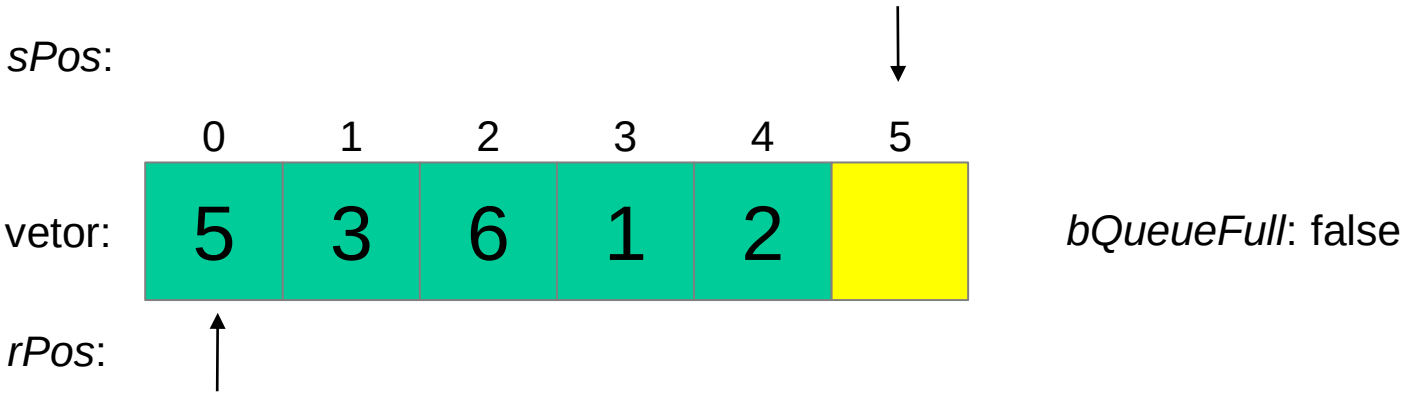



```
int frente;
:
:
frente = front( );

//frente: 5
```

Consulta a frente

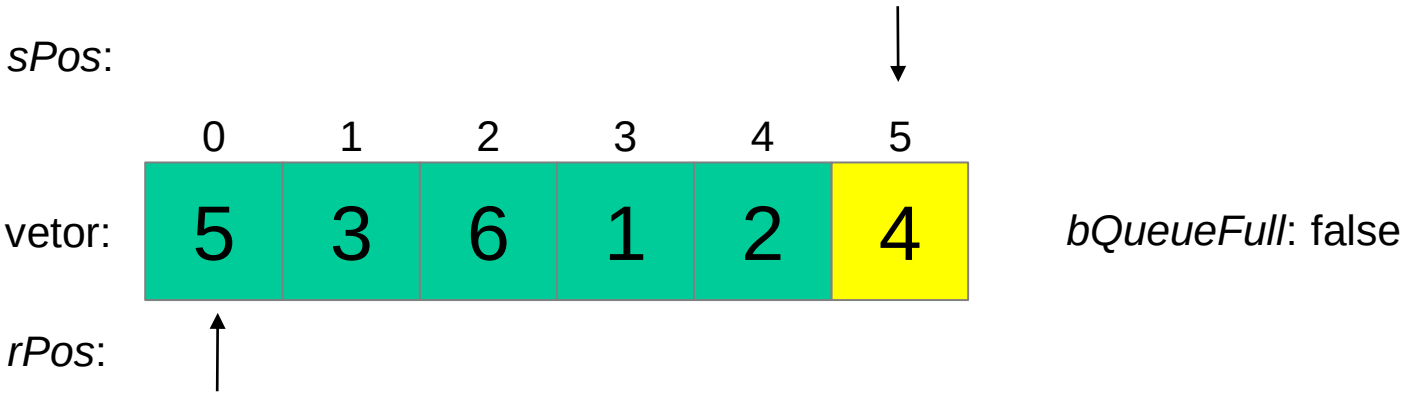
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(4);
```

Verifica se está cheia

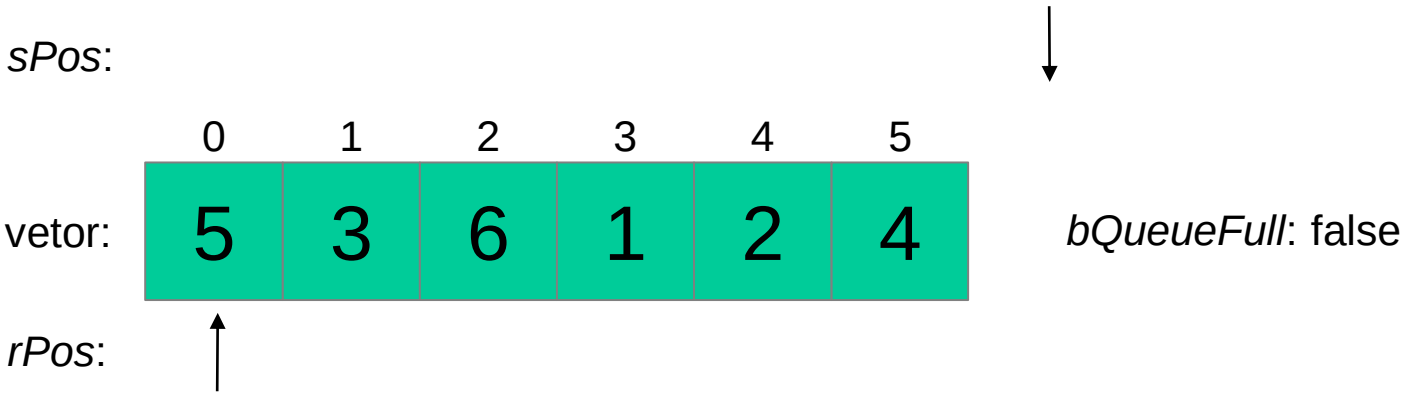
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(4);
```

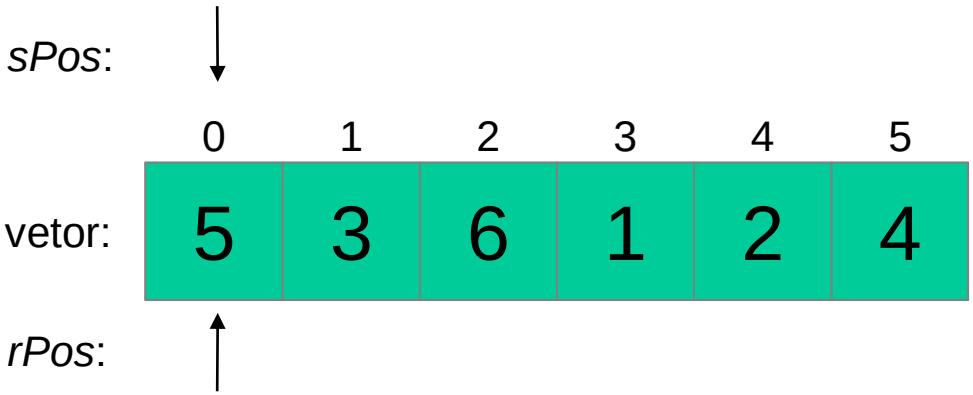
Não está cheia! Coloca o elemento 4

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

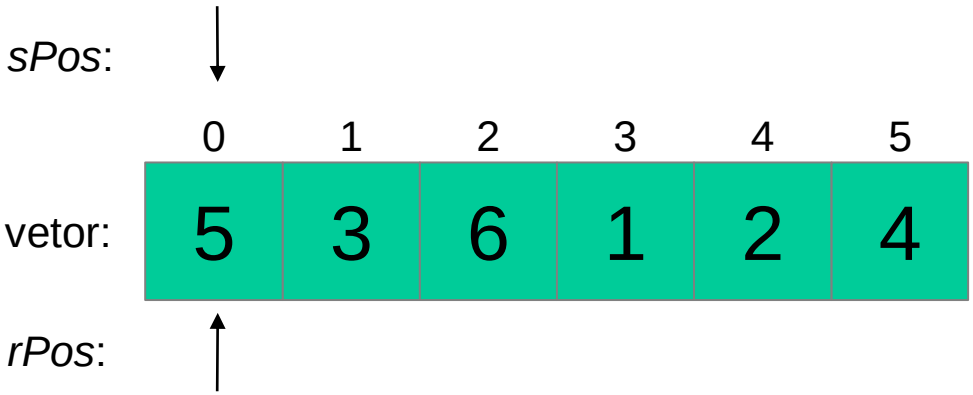
Fila Circular (Queue-C) – FIFO




```
if ( sPos >= vetor.length )
    sPos = 0;
else
    sPos++;
```

Ajusta o indicador de posição de inserção

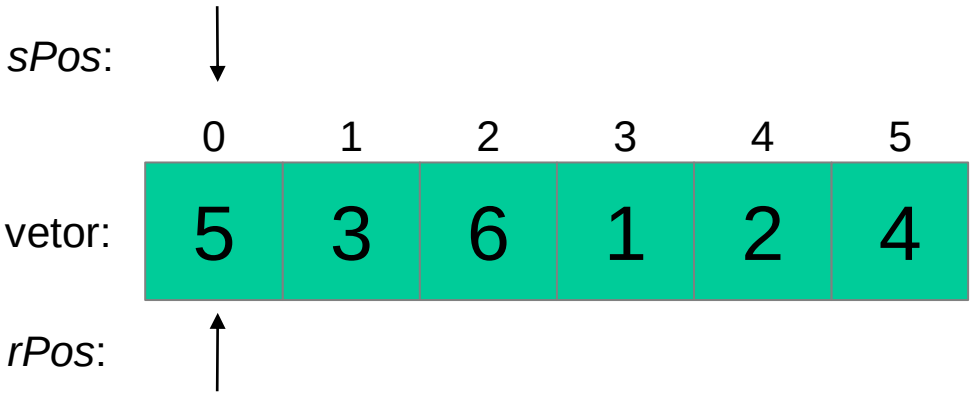
Fila Circular (Queue-C) – FIFO




```
if ( sPos == rPos )
    bQueueFull = true;
```

Indica que a fila circular está cheia

Fila Circular (Queue-C) – FIFO

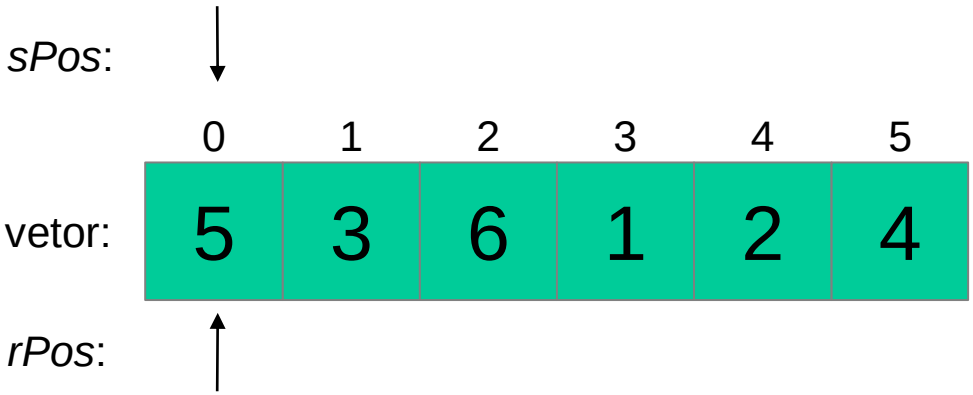



```
int frente;
:
:
frente = front( );

//frente: 5
```

Consulta a frente

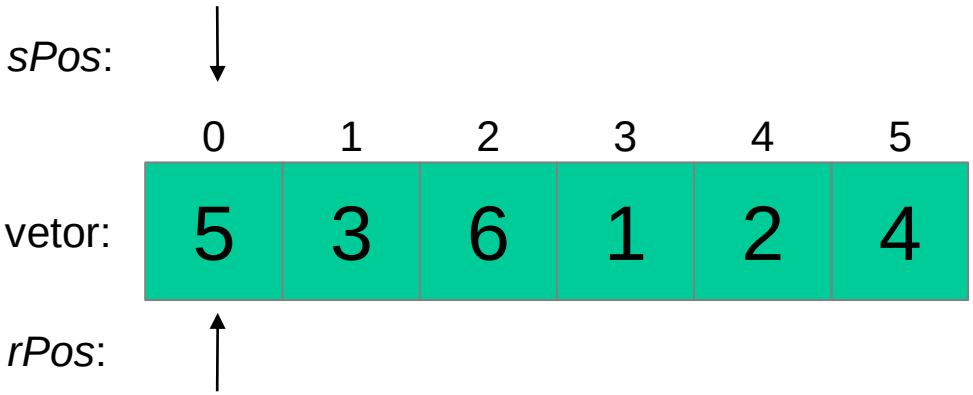
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(7);
```

Verifica se está cheia

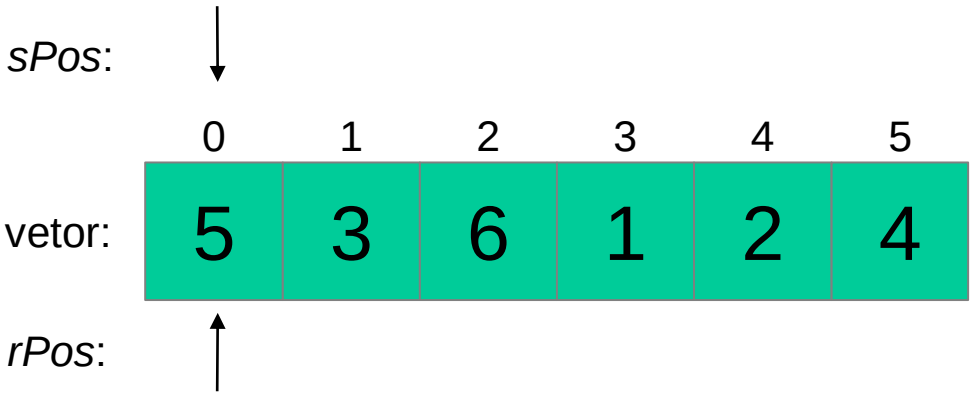
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(7);
```

Está cheia! Não coloca o elemento 7

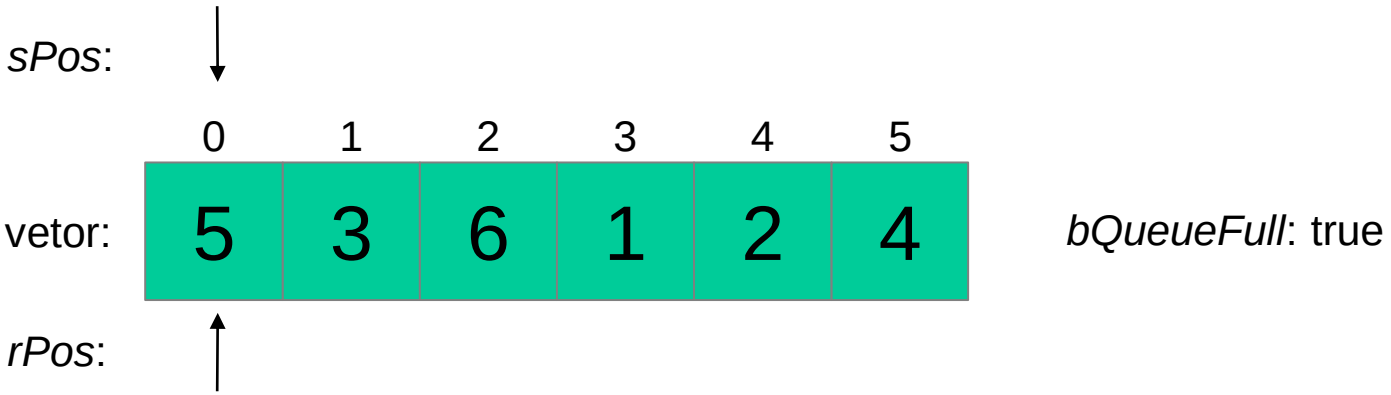
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Verifica se está vazia

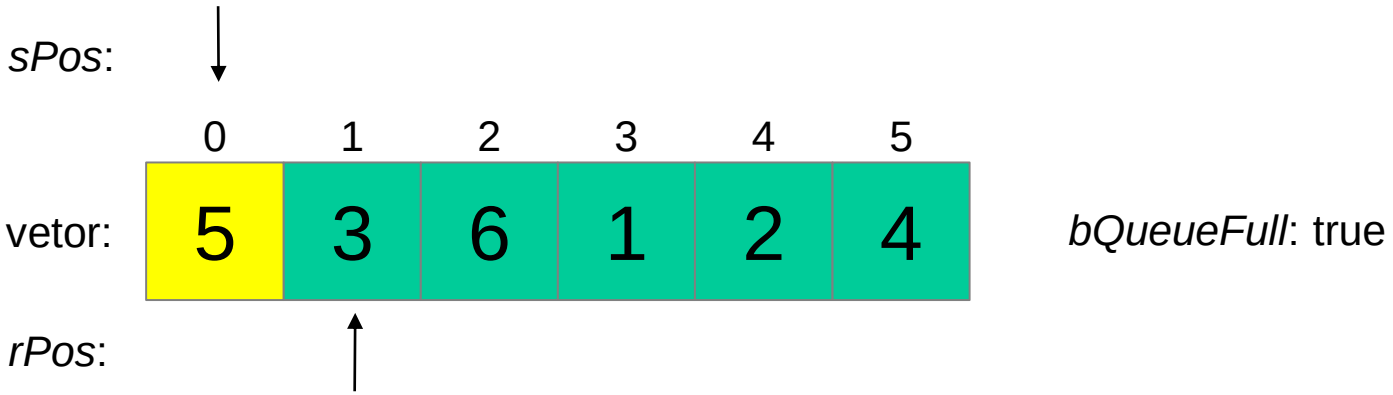
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
// n: 5
```

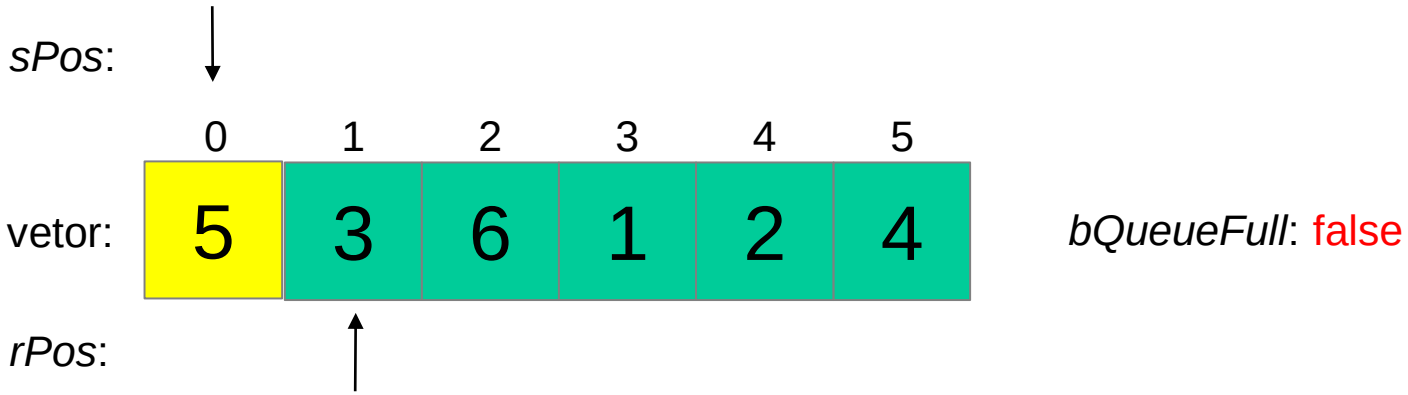
Não está vazia! Retira o elemento da frente

Fila Circular (Queue-C) – FIFO



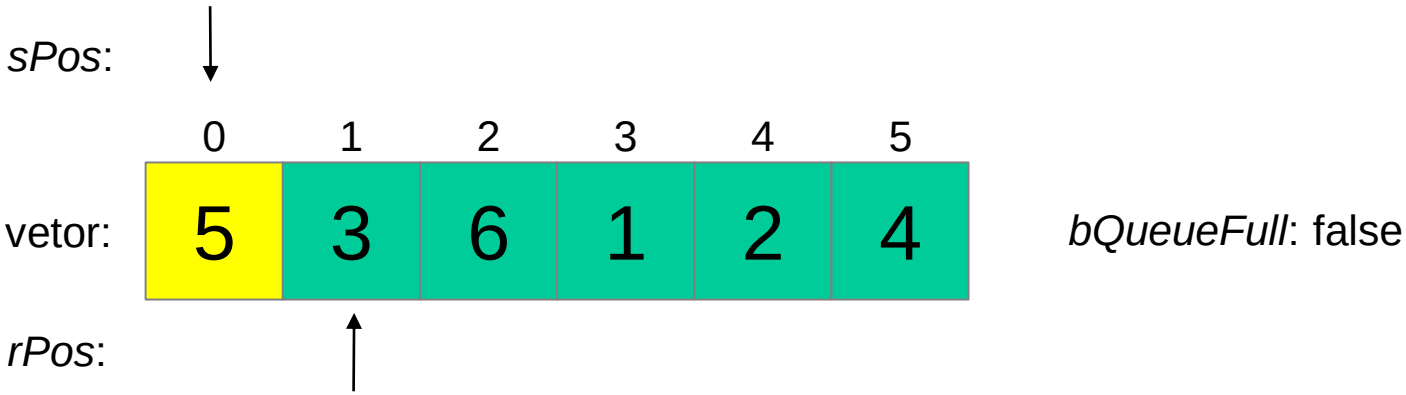
Incrementa indicador de posição de retirada

Fila Circular (Queue-C) – FIFO



Indica que a fila circular não está cheia

Fila Circular (Queue-C) – FIFO

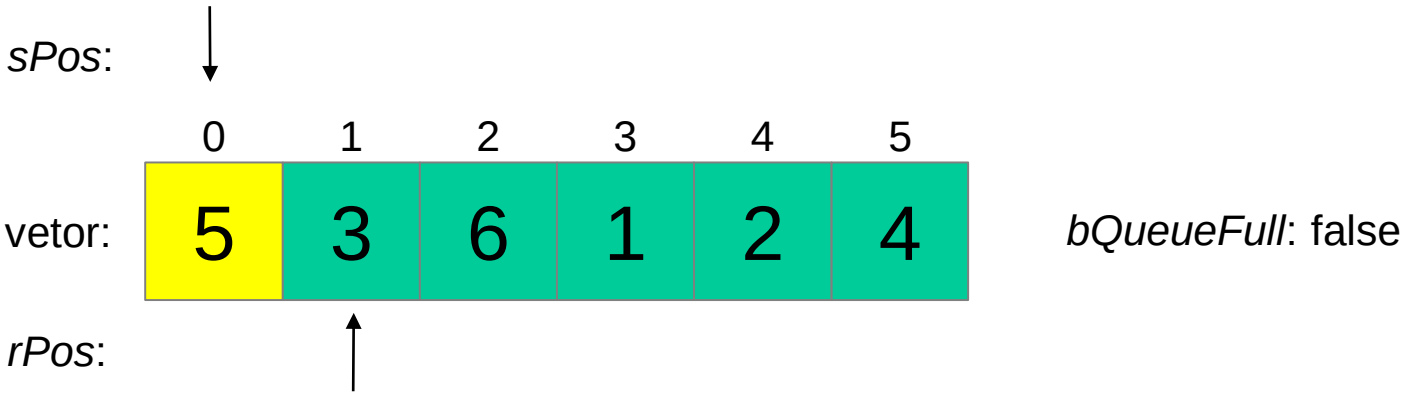



```
int frente;
:
:
frente = front( );

//frente: 3
```

Consulta a frente

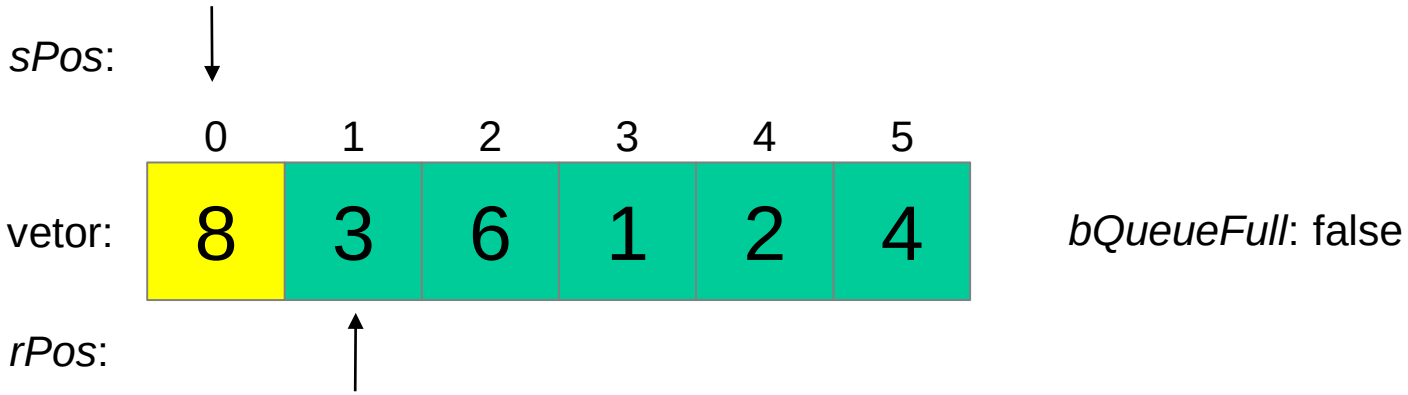
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(8);
```

Verifica se está cheia

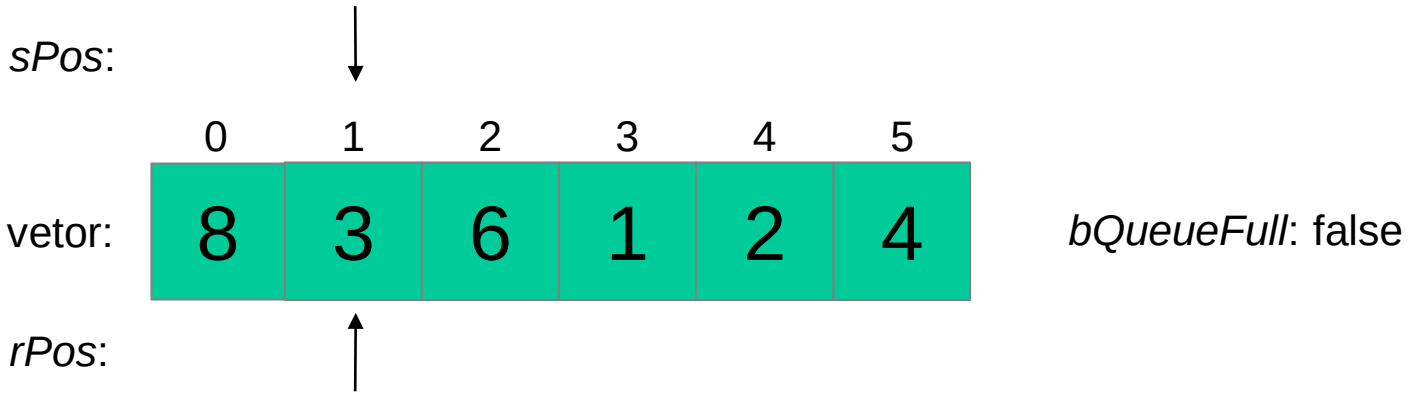
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(8);
```

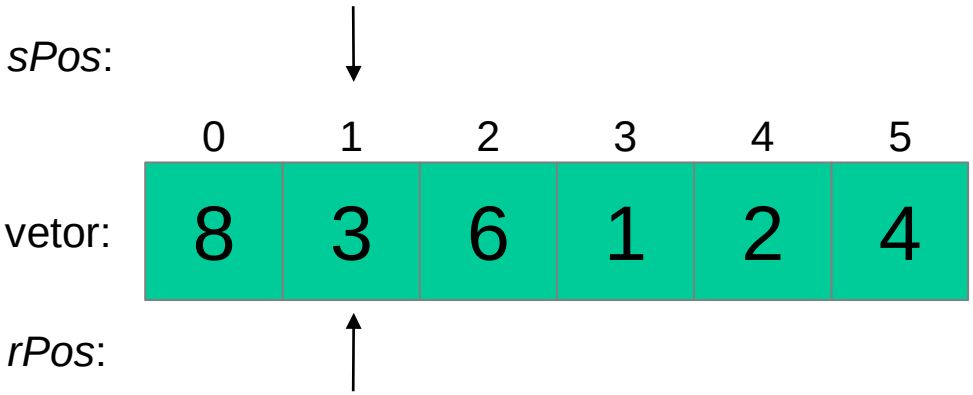
Não está cheia! Coloca o elemento 8

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

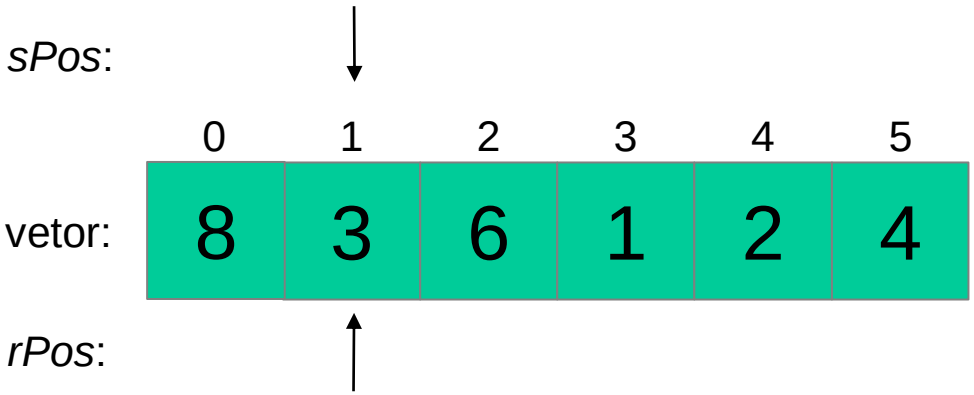
Fila Circular (Queue-C) – FIFO




```
if ( sPos == rPos )
    bQueueFull = true;
```

Indica que a fila circular está cheia

Fila Circular (Queue-C) – FIFO

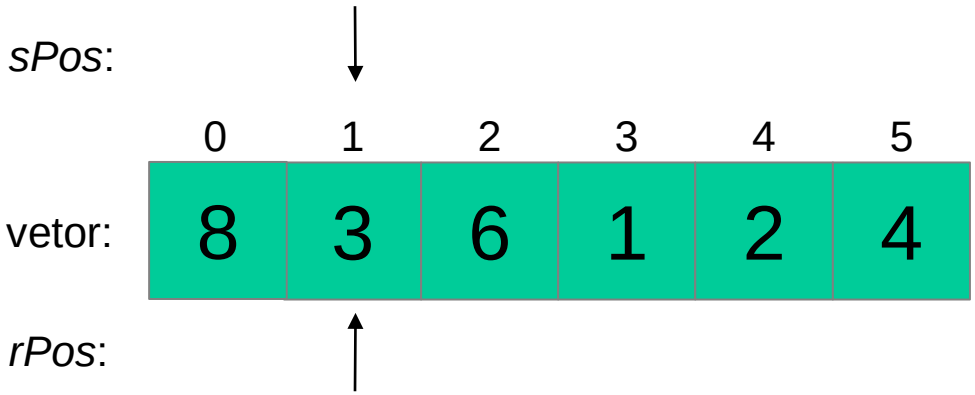


bQueueFull: false

```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Verifica se está vazia

Fila Circular (Queue-C) – FIFO

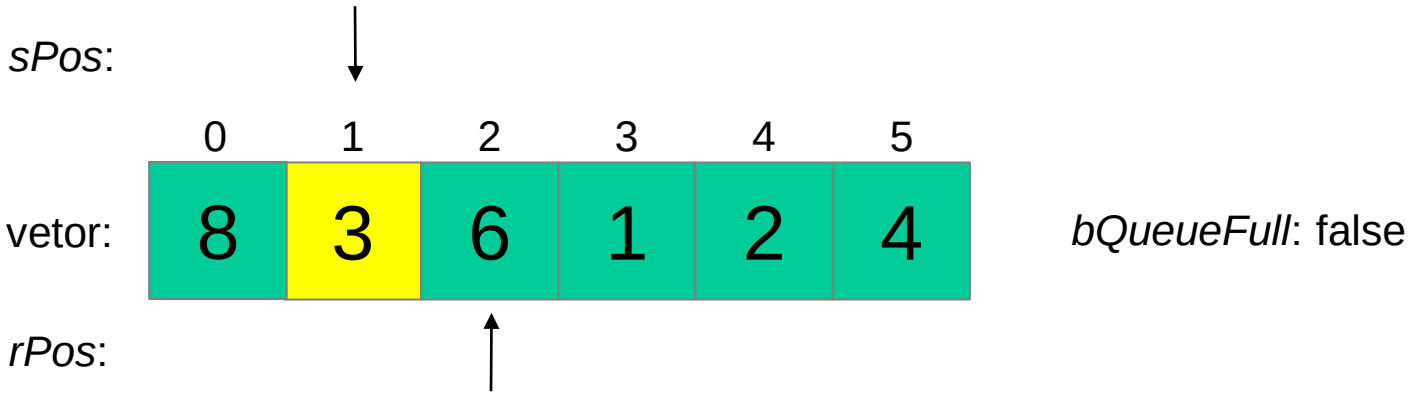


bQueueFull: false

```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
// n: 3
```

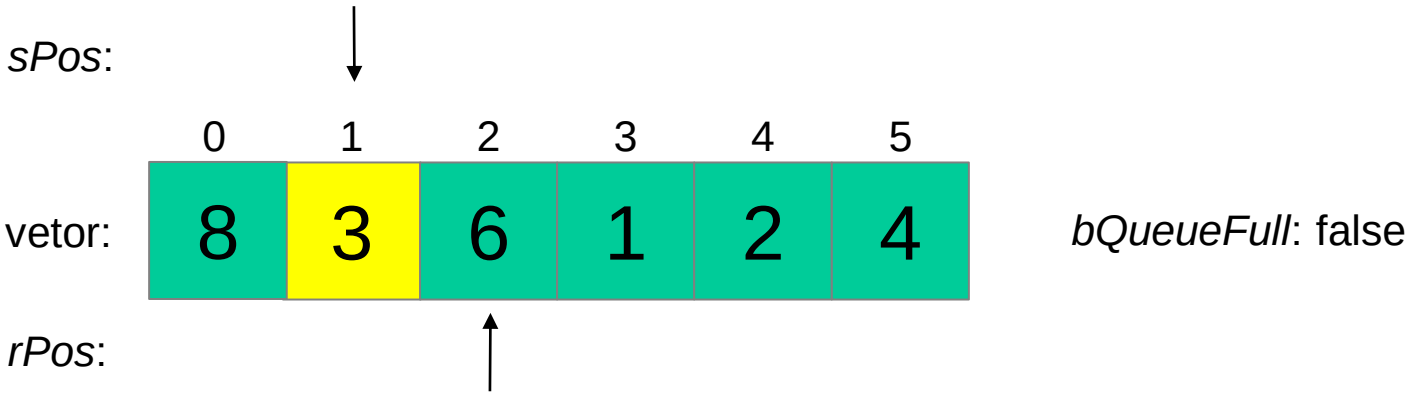
Não está vazia! Retira o elemento da frente

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de retirada

Fila Circular (Queue-C) – FIFO

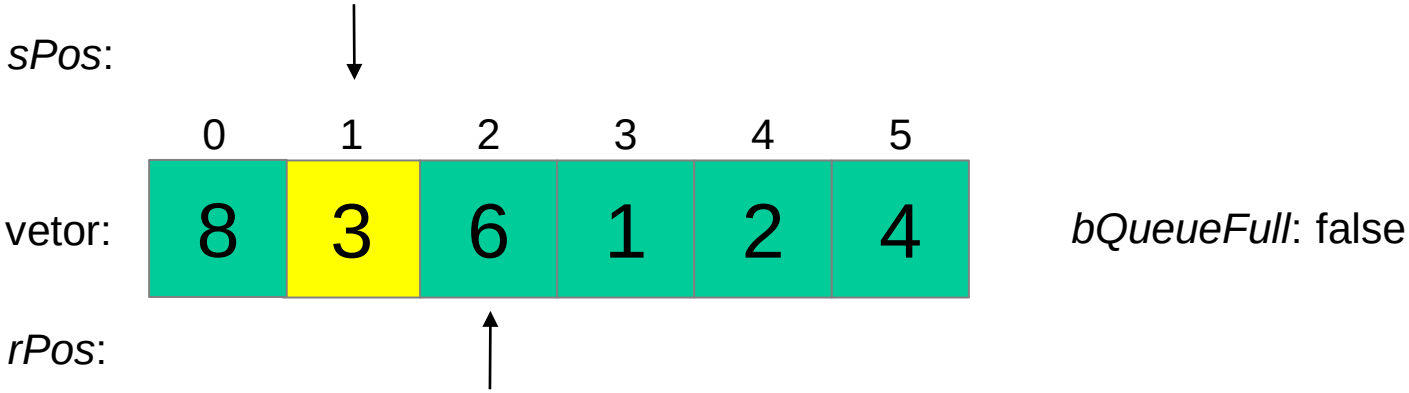



```
int frente;
:
:
frente = front( );

//frente: 6
```

Consulta a frente

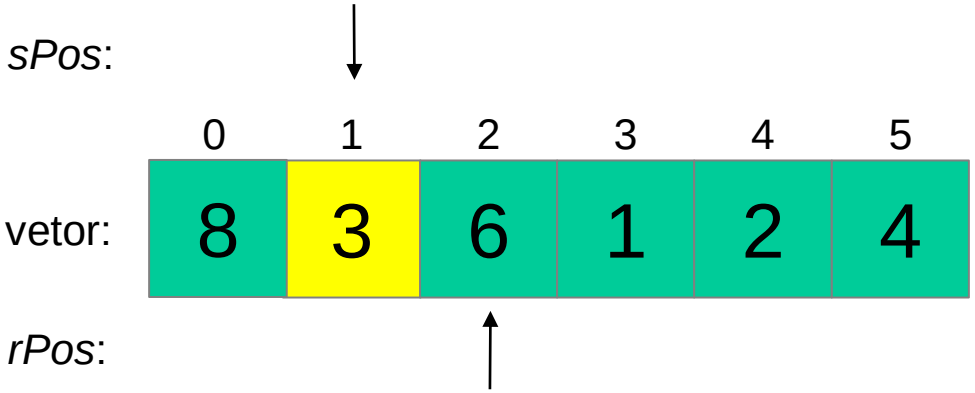
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println(“Fila Circular Vazia”);
else
       n = dequeueC( );
```

Verifica se está vazia

Fila Circular (Queue-C) – FIFO

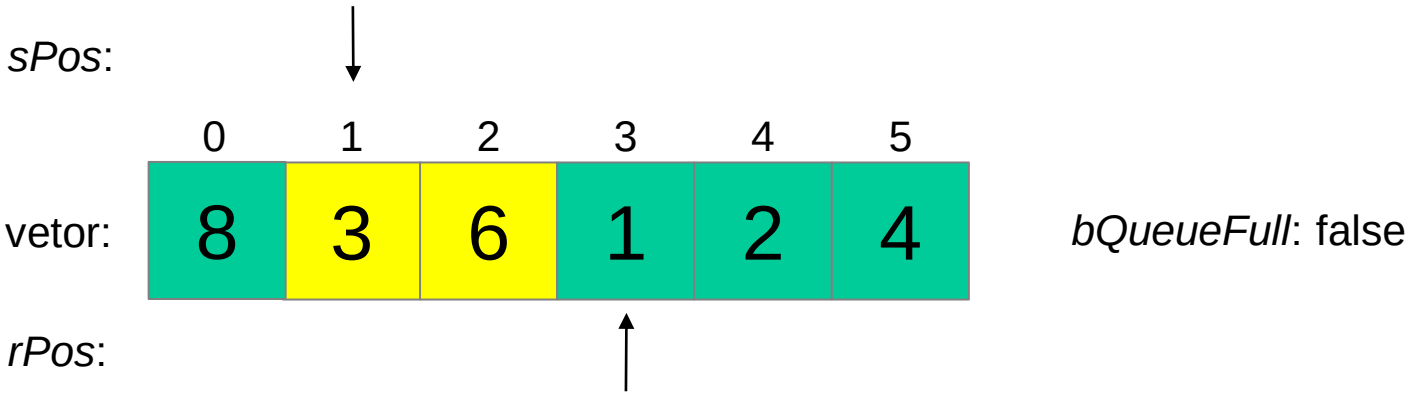


bQueueFull: false

```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
// n: 6
```

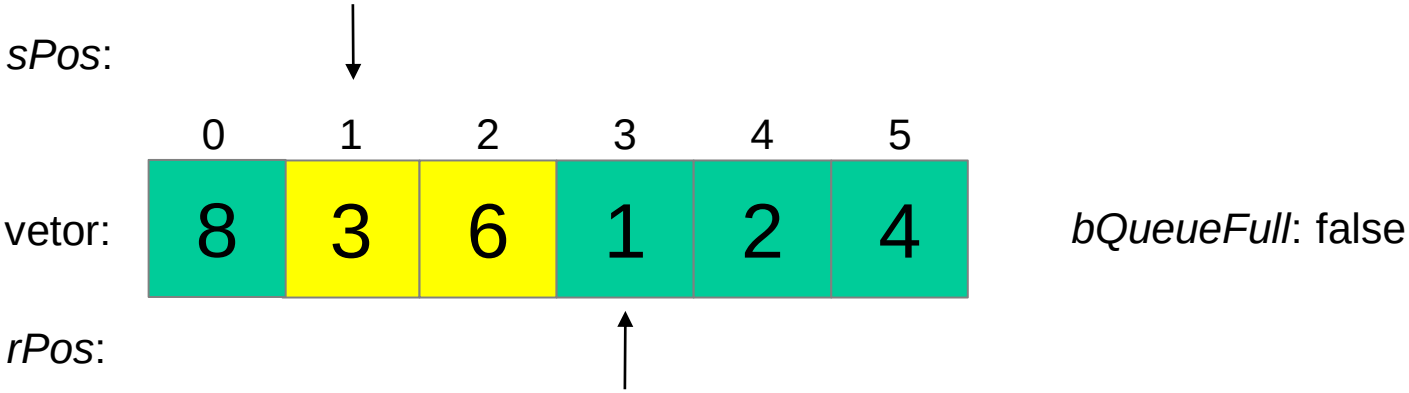
Não está vazia! Retira o elemento da frente

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de retirada

Fila Circular (Queue-C) – FIFO

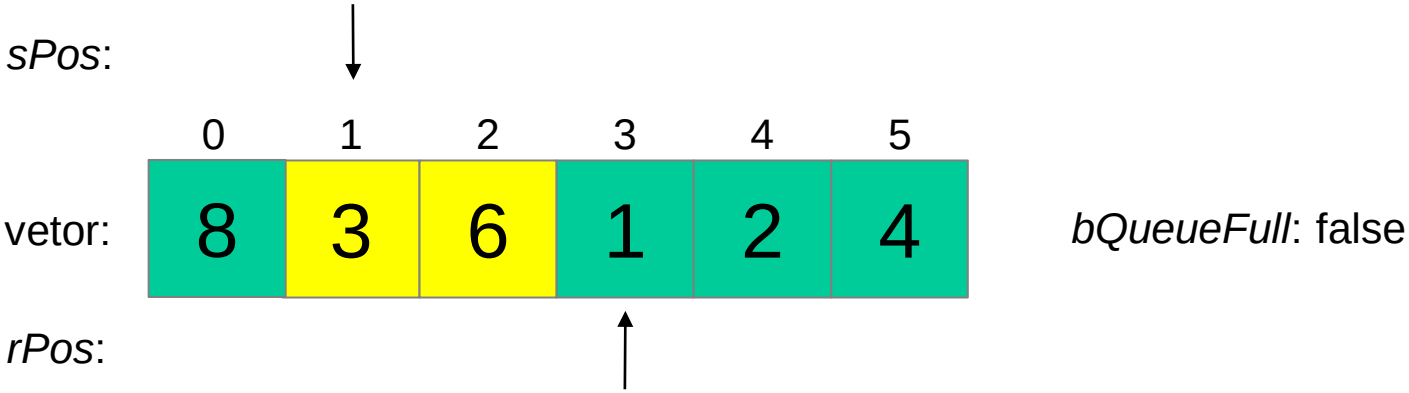



```
int frente;
:
:
frente = front( );

//frente: 1
```

Consulta a frente

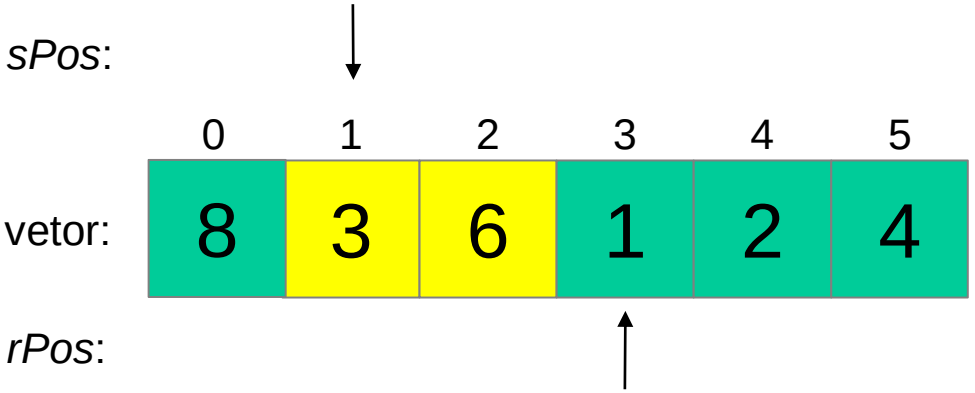
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Verifica se está vazia

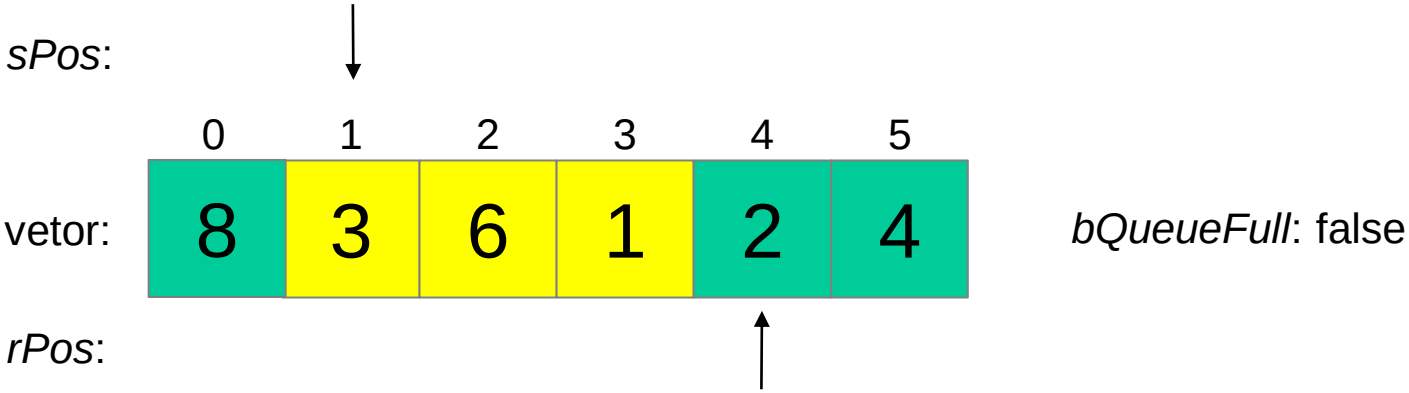
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
// n: 1
```

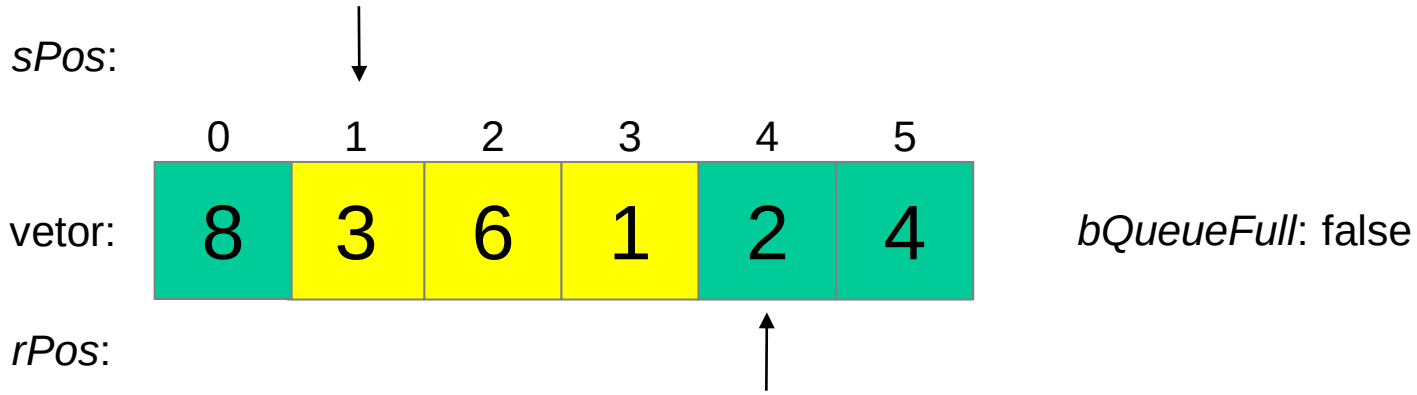
Não está vazia! Retira o elemento da frente

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de retirada

Fila Circular (Queue-C) – FIFO

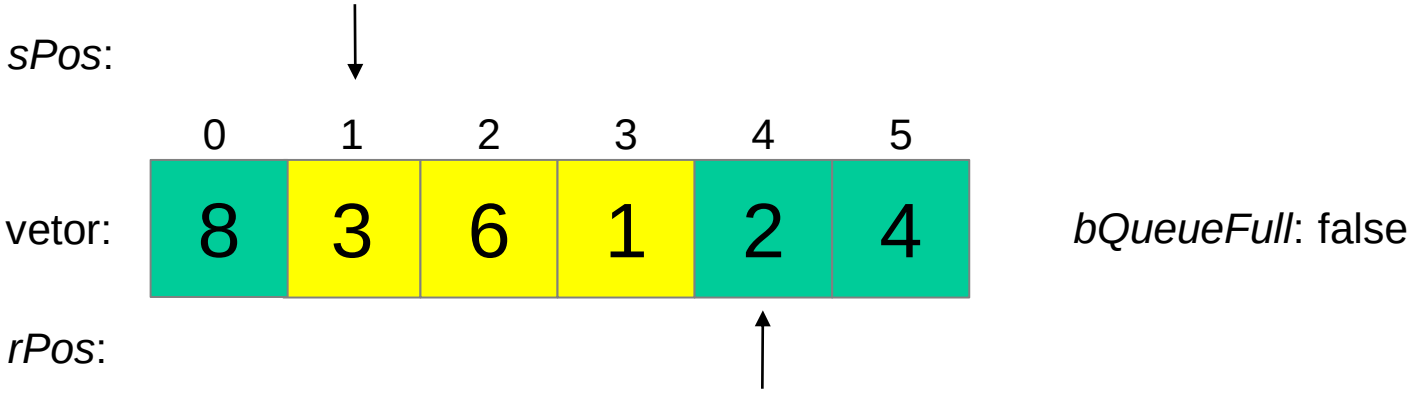



```
int frente;
:
:
frente = front( );

//frente: 2
```

Consulta a frente

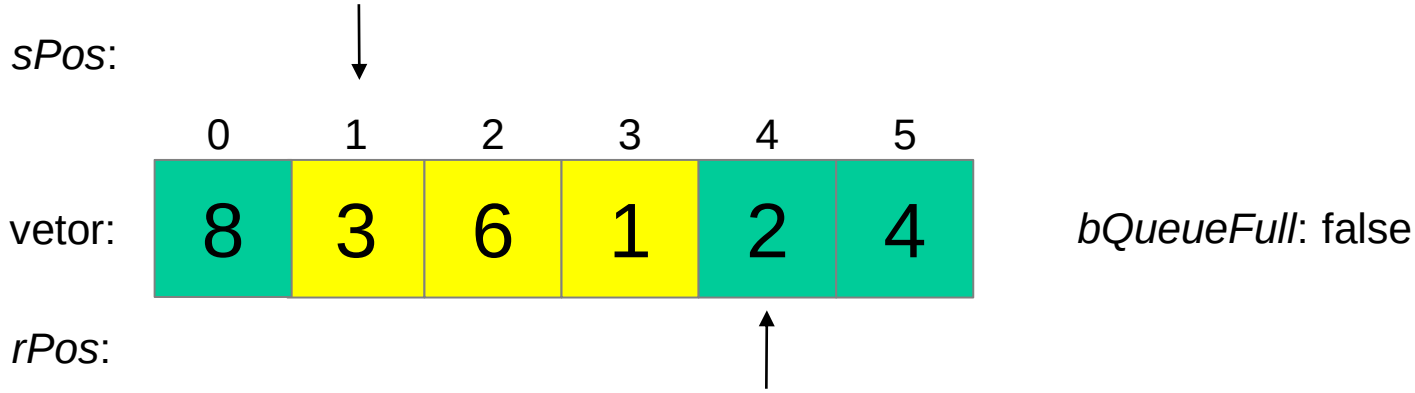
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Verifica se está vazia

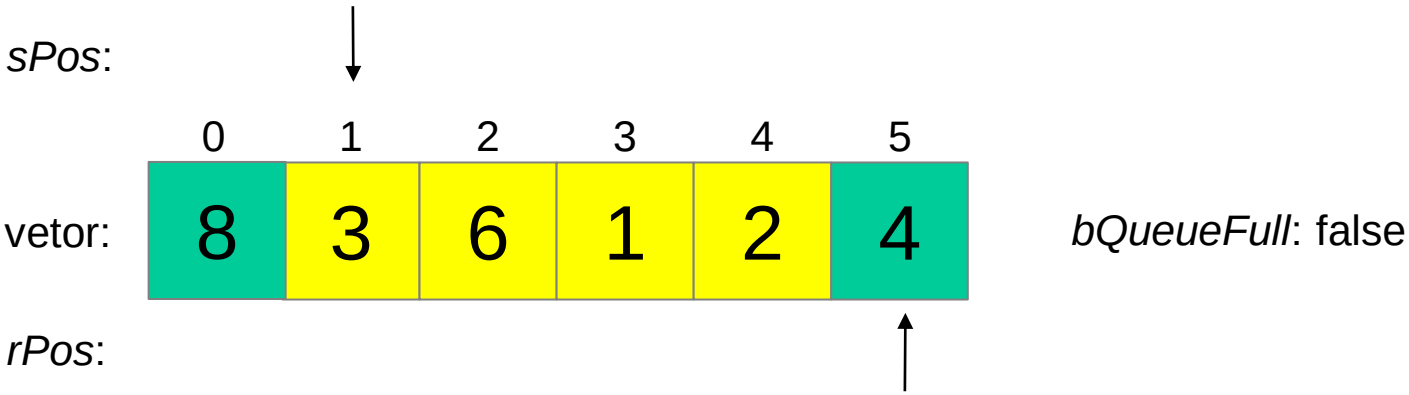
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
// n: 2
```

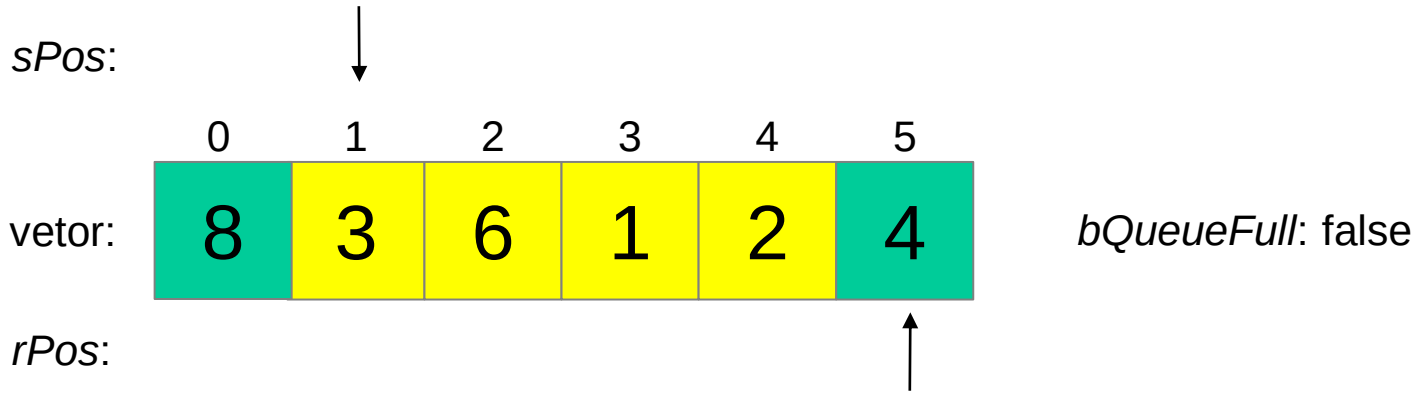
Não está vazia! Retira o elemento da frente

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de retirada

Fila Circular (Queue-C) – FIFO

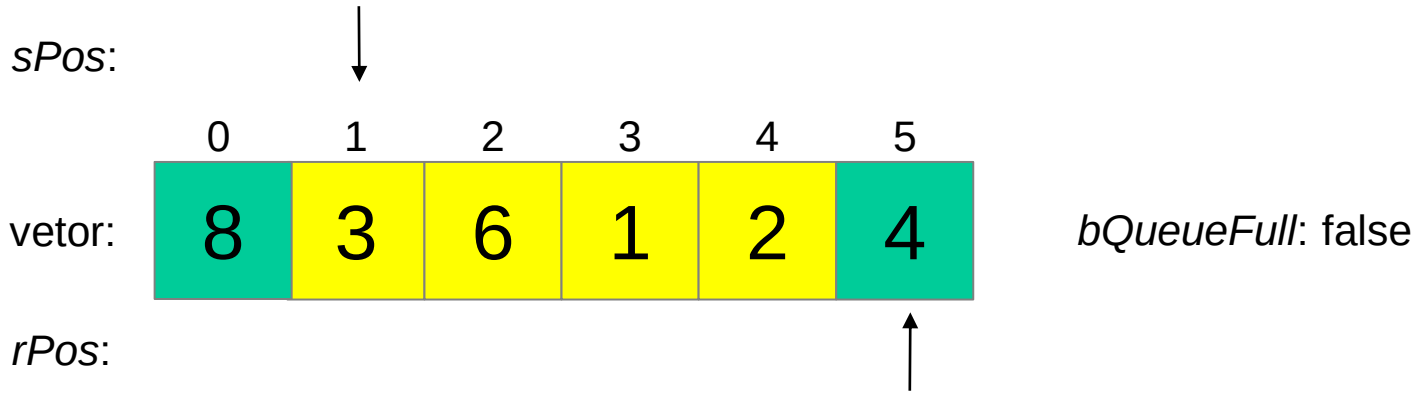



```
int frente;
:
:
frente = front( );

//frente: 4
```

Consulta a frente

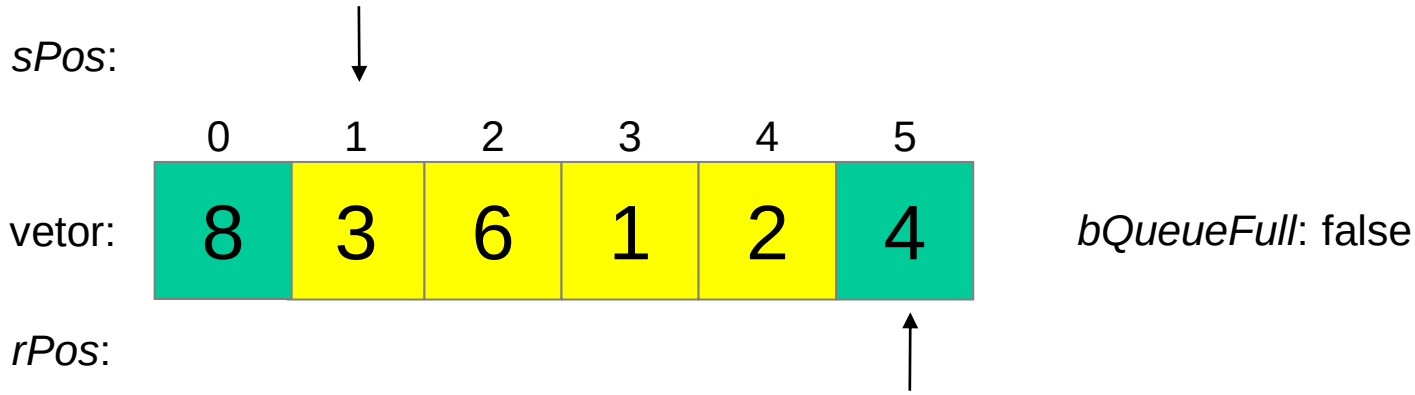
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Verifica se está vazia

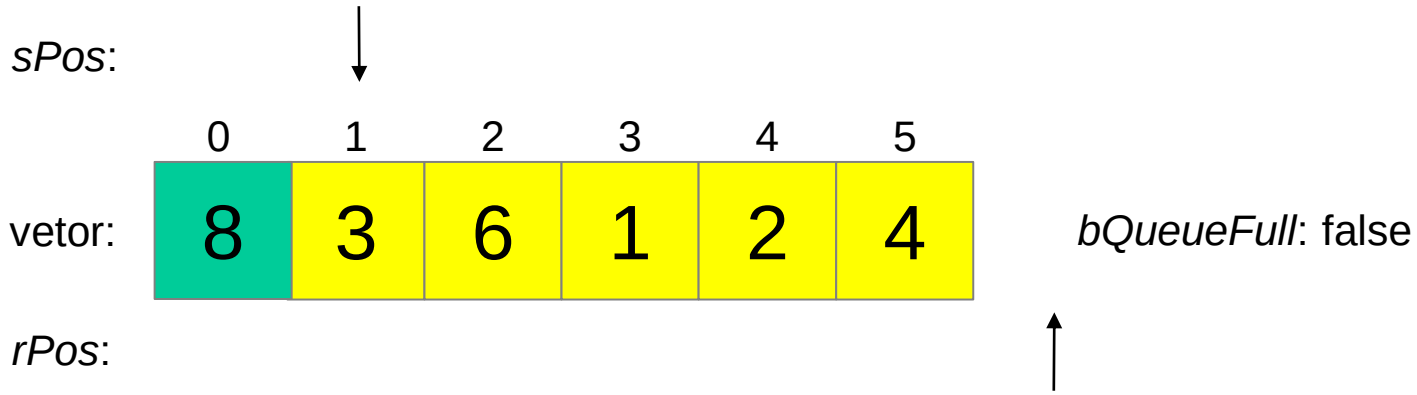
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
// n: 4
```

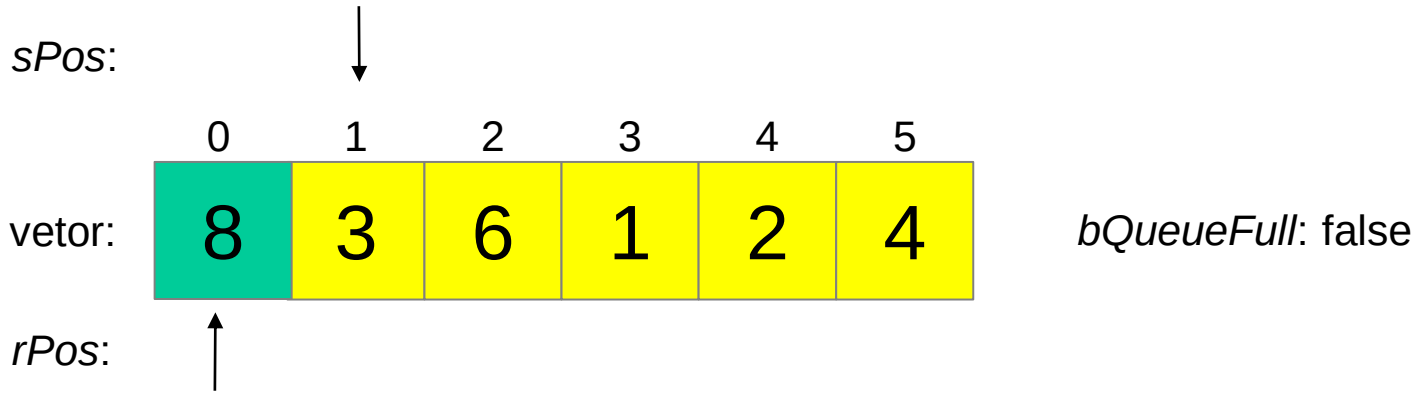
Não está vazia! Retira o elemento da frente

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de retirada

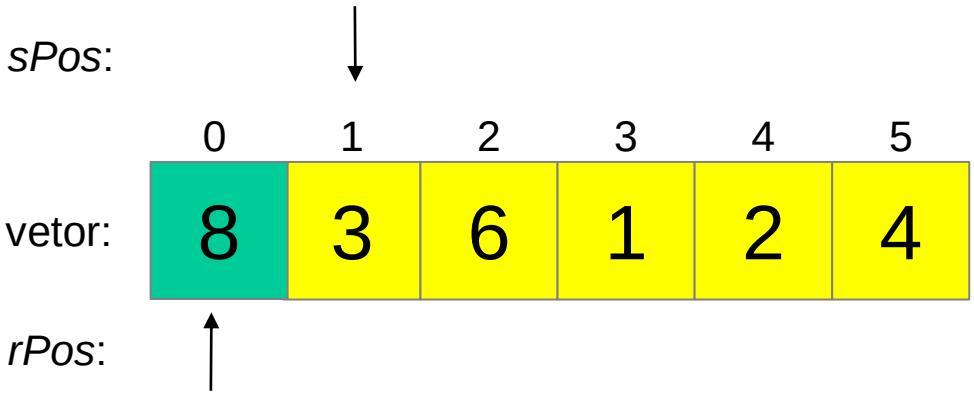
Fila Circular (Queue-C) – FIFO




```
if ( rPos >= vetor.length )
    rPos = 0;
else
    rPos++;
```

Ajusta o indicador de posição de retirada

Fila Circular (Queue-C) – FIFO



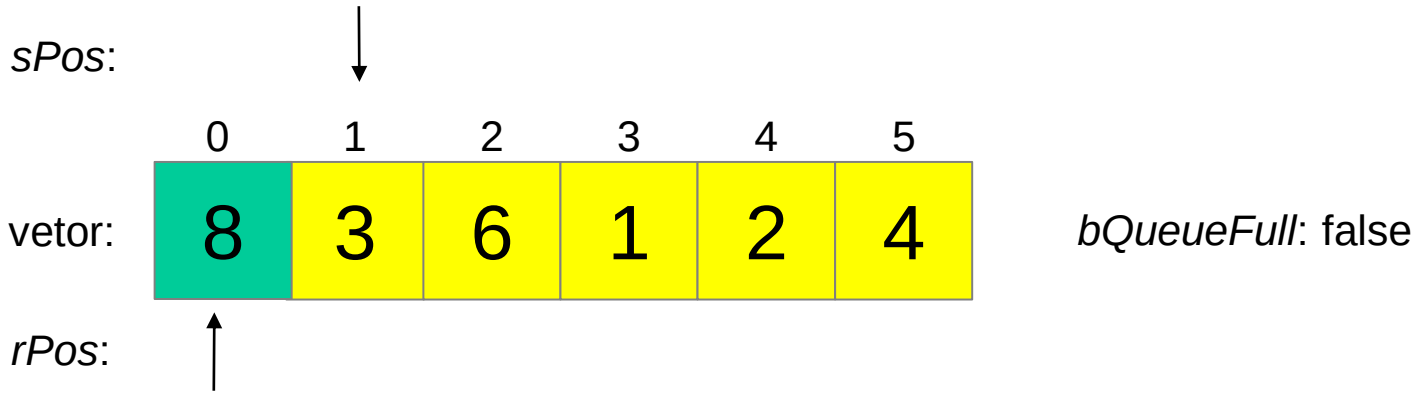
bQueueFull: false

```
int frente;
:
:
frente = front( );

//frente: 8
```

Consulta a frente

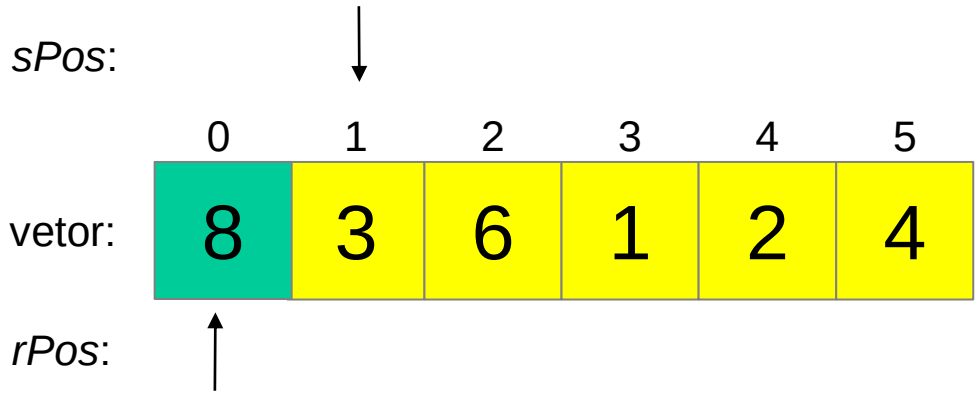
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Verifica se está vazia

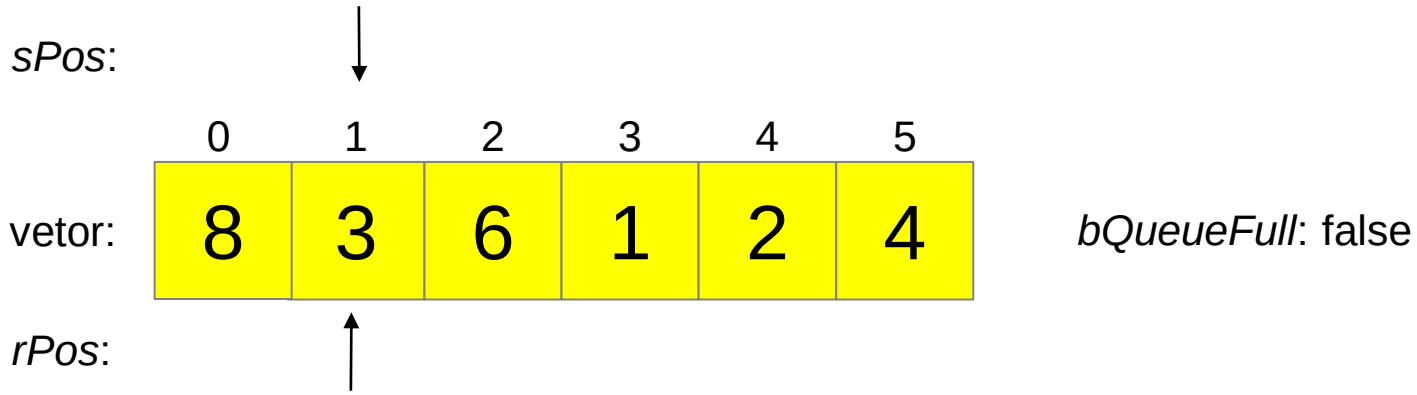
Fila Circular (Queue-C) – FIFO




```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
// n: 8
```

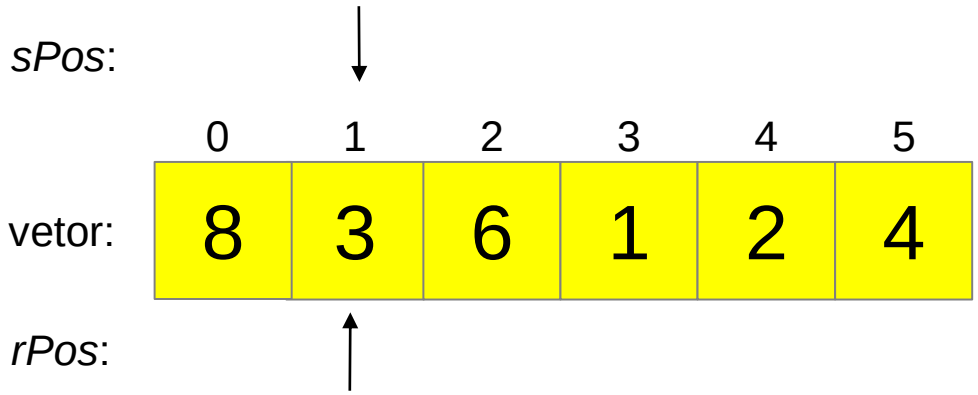
Não está vazia! Retira o elemento da frente

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de retirada

Fila Circular (Queue-C) – FIFO

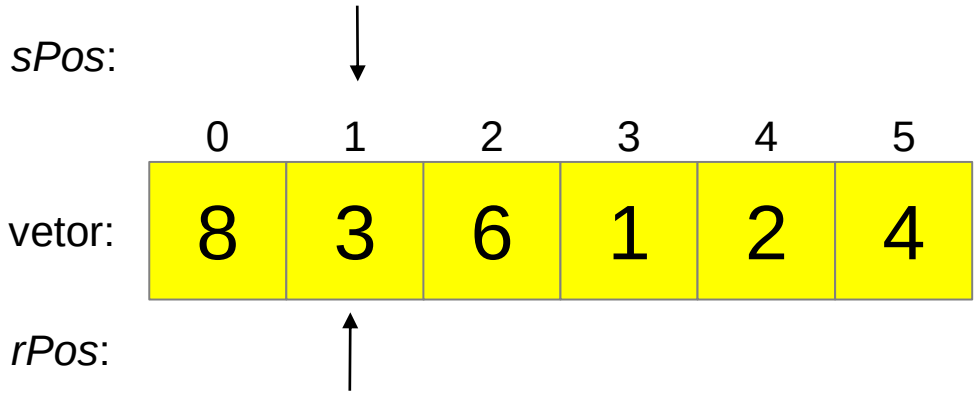


bQueueFull: false

```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Verifica se está vazia

Fila Circular (Queue-C) – FIFO

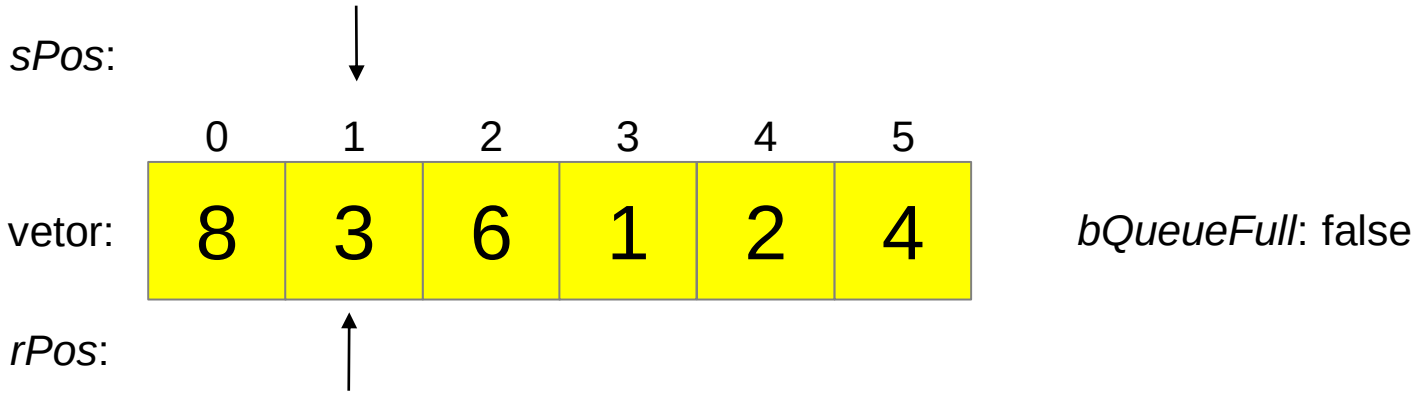


bQueueFull: false

```
int n;
:
:
if ( size( ) <= 0 )
      System.out.println("Fila Circular Vazia");
else
       n = dequeueC( );
```

Está vazia! Não retira o elemento da frente

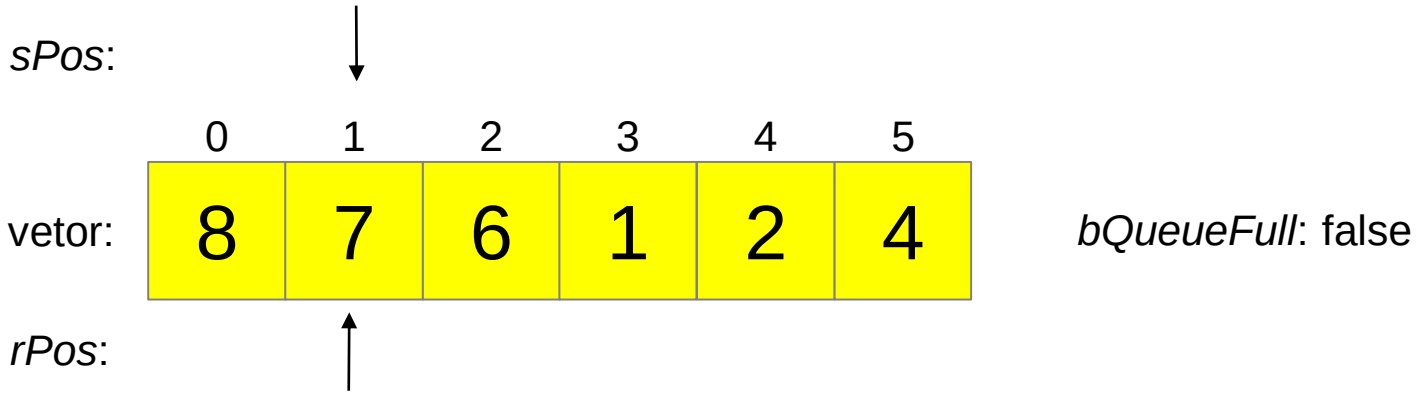
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(7);
```

Verifica se está cheia

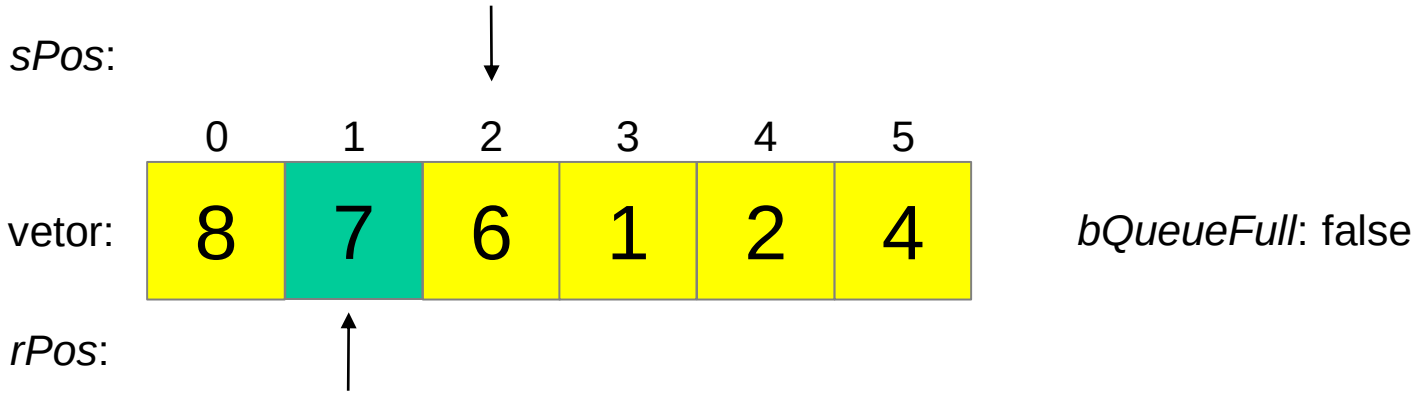
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(7);
```

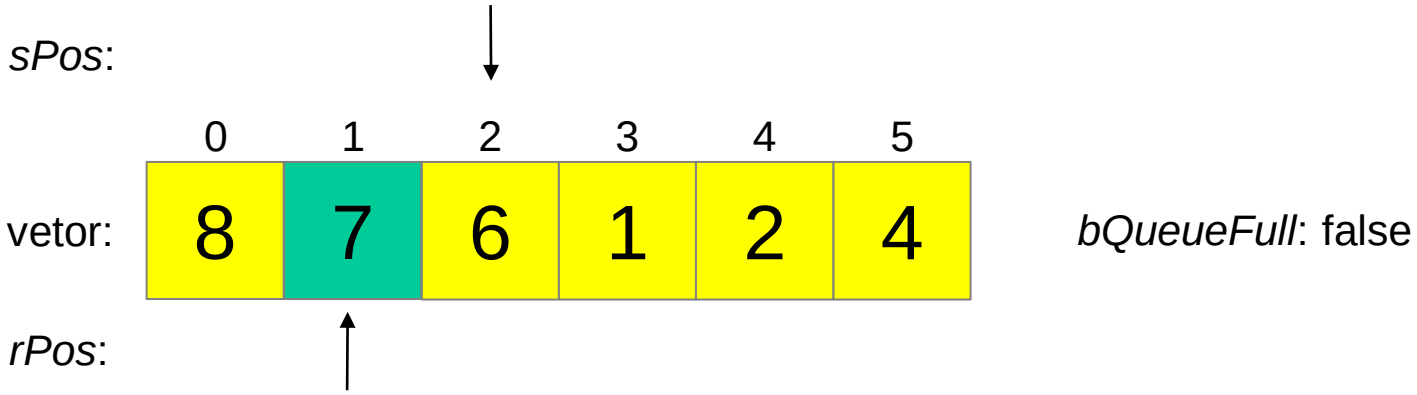
Não está cheia! Coloca o elemento 7

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

Fila Circular (Queue-C) – FIFO

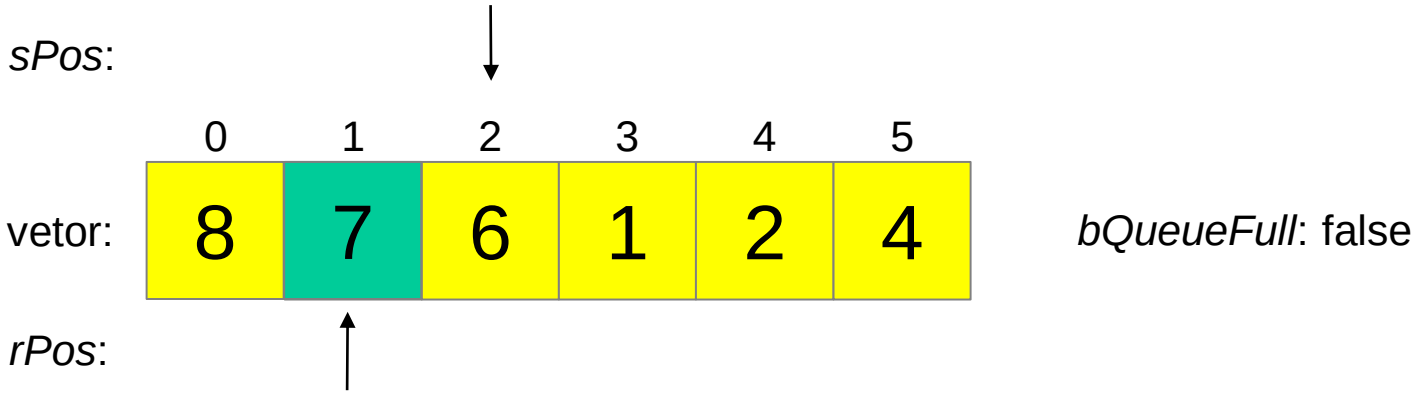



```
int frente;
:
:
frente = front( );

//frente: 7
```

Consulta a frente

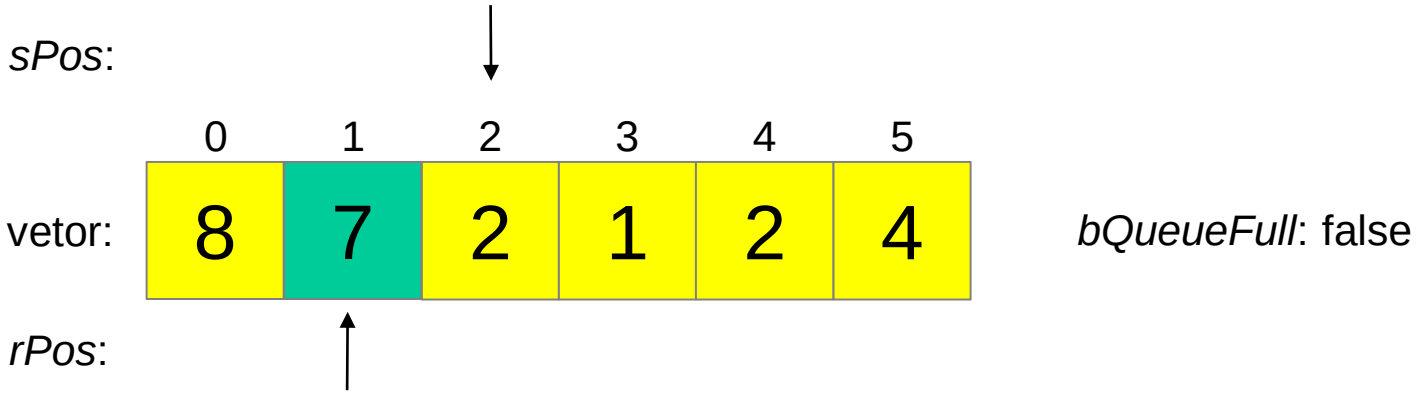
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(2);
```

Verifica se está cheia

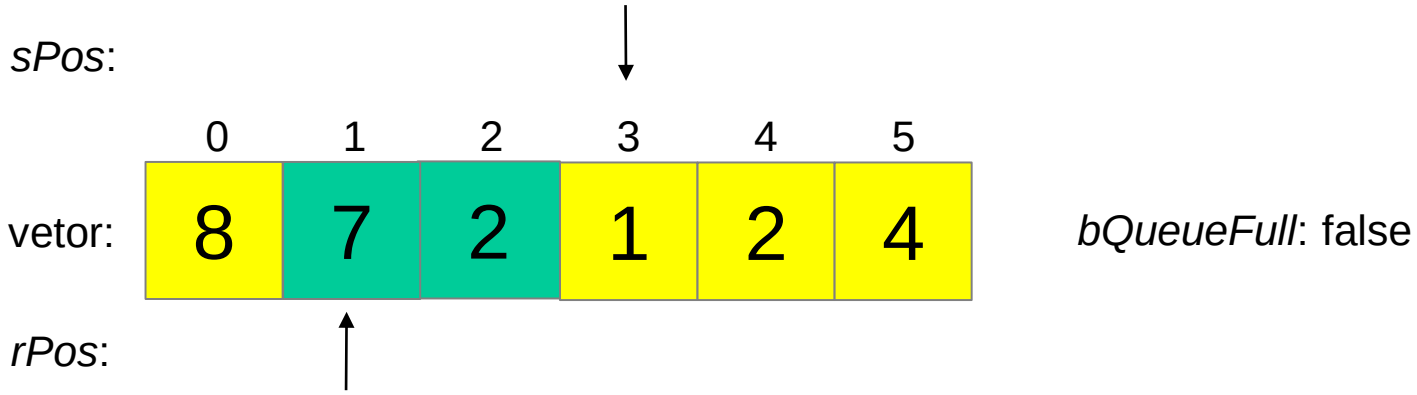
Fila Circular (Queue-C) – FIFO




```
if ( isOver( ) )
    System.out.println("Fila Circular Cheia");
else
    enqueueC(2);
```

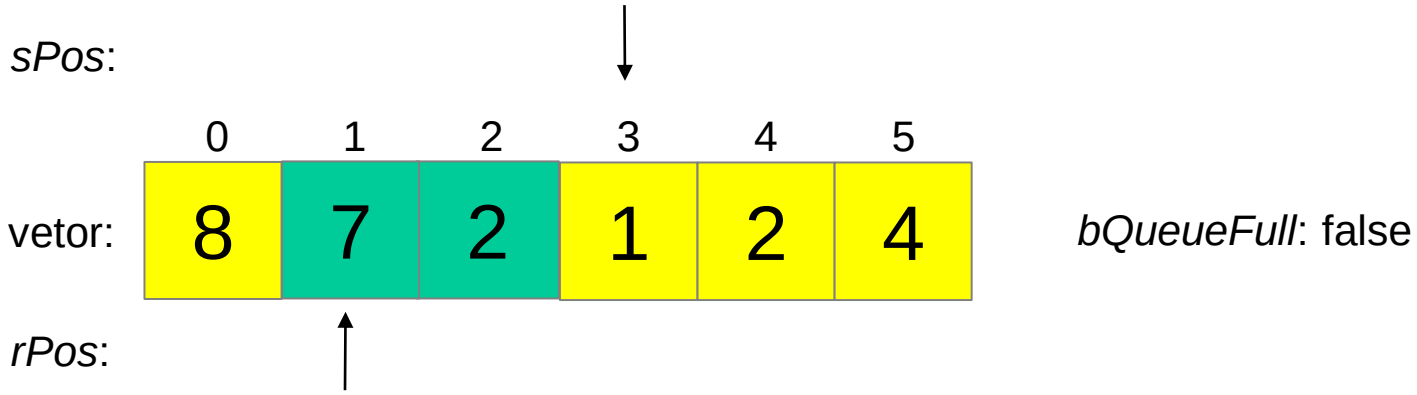
Não está cheia! Coloca o elemento 2

Fila Circular (Queue-C) – FIFO



Incrementa indicador de posição de inserção

Fila Circular (Queue-C) – FIFO

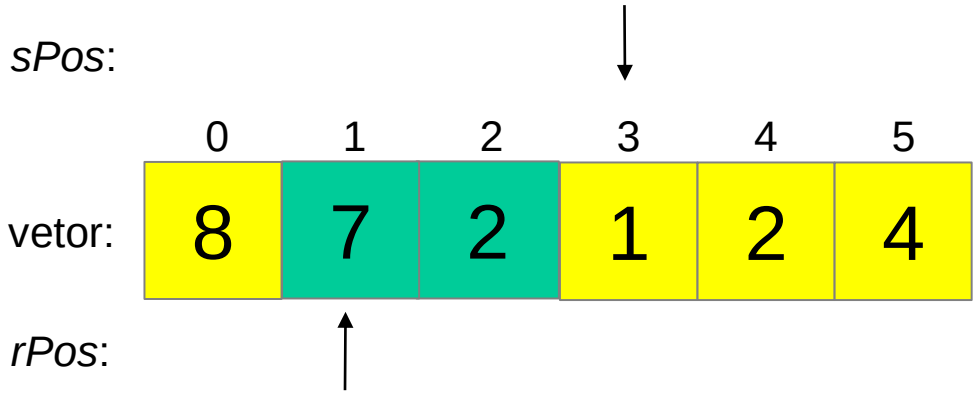



```
int frente;
:
:
frente = front( );

//frente: 7
```

Consulta a frente

Fila Circular (Queue-C) – FIFO



bQueueFull: false

Repete qualquer operação:
enqueueC, *dequeueC*, *size* e *front*,
conforme necessidade...

Insere ou retira elementos da fila...

Fila Circular (Queue-C) – FIFO

Exemplo de códigos escrito em Java:

Codificação dos Métodos

Fila Circular (Queue-C) – FIFO

```
public   static    int          iTAM = 10;
public   static    int          iFila [ ] = new    int [ iTAM ];
public   static    int          iSPos = 0;
public   static    int          iRPos = 0;
public   static    boolean      bQueueFull = false;
:
:
public   static    int   size ( )
{
if ( iSPos >= iRPos  &&  ! bQueueFull )  return   iSPos - iRPos;
else     return   iSPos + iFila.length - iRPos;
}

public   static    int   front ( )
{
     return   iFila [ iRPos ];
}
```

Métodos *size( )* e *front( )*

Fila Circular (Queue-C) – FIFO

```
public   static    int          iTAM = 10;
public   static    int          iFila [ ] = new    int [ iTAM ];
public   static    int          iSPos = 0;
public   static    int          iRPos = 0;
public   static    boolean      bQueueFull = false;

public   static    boolean      isOver ( )
{
if ( iSPos == iRPos  &&  bQueueFull )    return    true;
return   false;
}
public   static    void         enQueueC ( int iC )
{
iFila [ iSPos++ ] = iC;
if ( iSPos >= iFila.length )    iSPos=0;
if( iSPos == iRPos)  bQueueFull = true;
}
```

Métodos *isOver( )* e *enQueueC( )*

Fila Circular (Queue-C) – FIFO

```
public   static    int       iTAM = 10;
public   static    int       iFila [ ] = new    int [ iTAM ];
public   static    int       iSPos = 0;
public   static    int       iRPos = 0;

:
:

public   static    int   deQueueC ( )
{
int   iIndice = iRPos++;

if ( iRPos >= iFila.length )    iRPos = 0;

bQueueFull = false;
return   iFila [ iIndice ];
}
```

Método *deQueueC( )*

Fila Circular (Queue-C) – FIFO

FIM