CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Filas

Disciplina: Estrutura de Dados I

Prof. Fermín Alfredo Tang Montané

Curso: Ciência da Computação

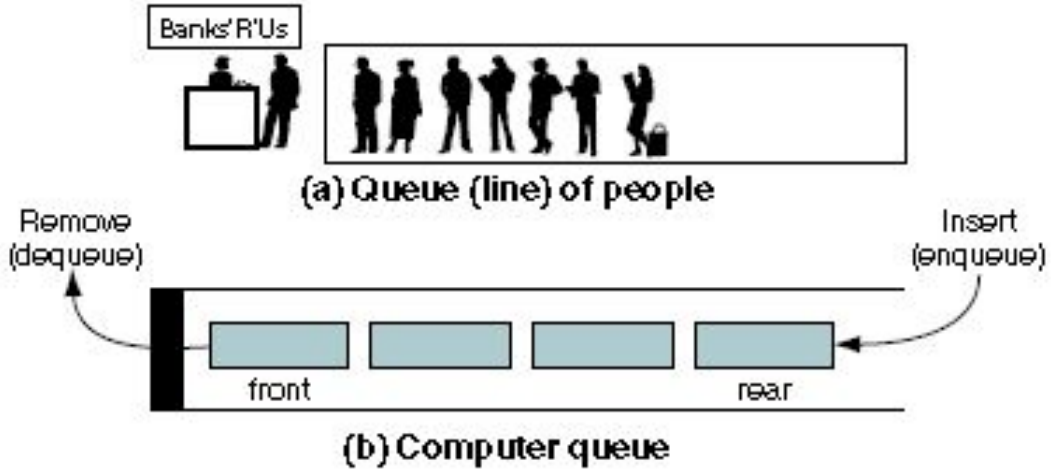
Filas (Queues)
Definição

- Um fila (queue) é uma lista linear na qual os dados somente podem ser inseridos em uma extremidade chamada de fim da fila (rear), e removidos da outra extremidade chamada de inicio da fila (front).
- Essas restrições garantem que os dados que se encontram na fila serão processados na ordem em que eles foram recebidos.
- Ao inserir uma sequência de dados em uma fila e depois remover essa sequência, a ordem desses dados permanecerá a mesma.
- Exemplo: Os dados inseridos na sequência {5, 10, 15, 20} serão removidos como {5, 10, 15, 20}.
- O atributo (ou caraterística) de preservação da ordem é motivo pelo qual a fila é conhecida como uma estrutura de dados FIFO (*First in – First Out*), primeiro a entrar, primeiro a sair.

Filas (Queues)

Exemplos

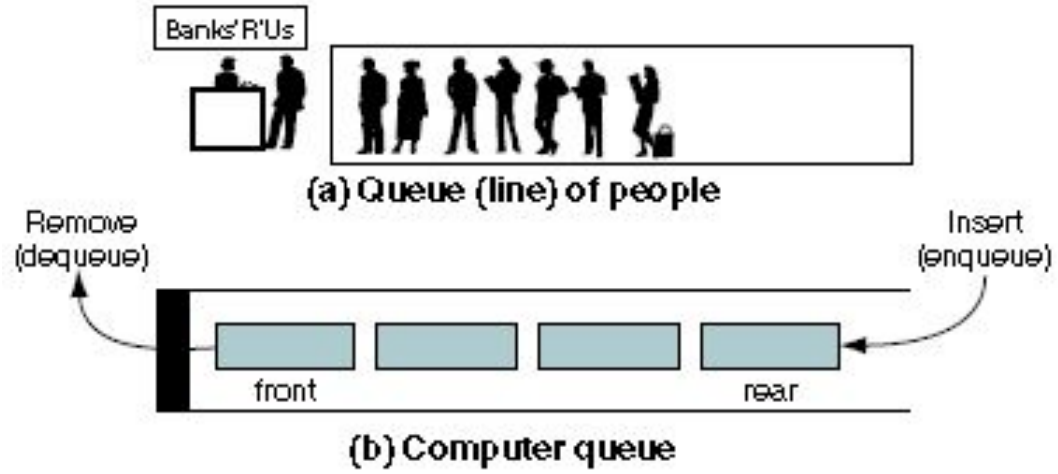
- O conceito de fila também é muito familiar. Utilizamos e encontramos diferentes tipos de filas no nosso dia a dia. Por exemplo:
  - Filas de pessoas esperando pelo ônibus;
  - Lista de chamadas colocadas em espera para serem respondidas por um operador telefónico;
  - Uma lista de trabalhos em espera para serem processados por um computador. Fila de impressão.
- Qualquer situação em que se preserve a ordem dos elementos é uma fila.



Filas (Queues)

Exemplos

- Qualquer situação em que se preserve a ordem dos elementos é uma fila.
- Na figura, os elementos, sejam pessoas ou dados, entram na fila pelo fim dela e progridem até chegar ao inicio da fila. Uma vez atingido o inicio da fila, deixam a fila e são atendidos.



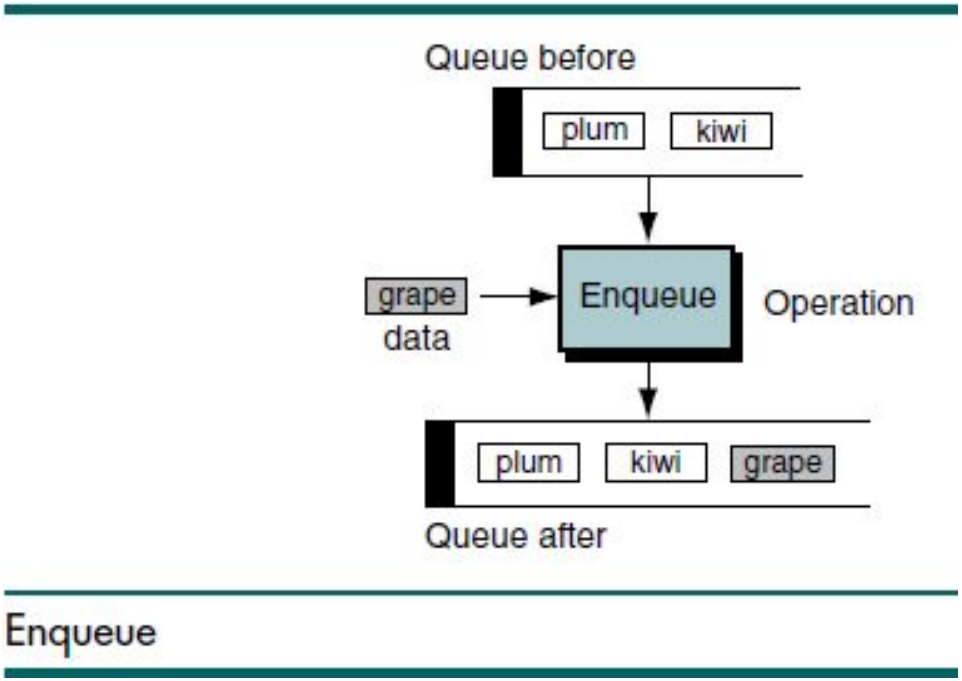
- Um fila (queue) é uma estrutura de dados FIFO (*First in – First Out*) em que as inserções acontecem em uma extremidade chamada Fim da Fila (rear) e as remoções acontecem na outra extremidade chamada Inicio da Fila (front).

Filas (Queues)
Operações Básicas

- Existem quatro operações básicas de filas:
  - Enfileirar (Inserir Fila) (enqueue)
  - Desenfileirar (Remover Fila) (dequeue)
  - Recuperar Inicio Fila (queue front)
  - Recuperar Fim Fila (queue rear)

Filas (Queues)
Operação Enfileirar (Enqueue)

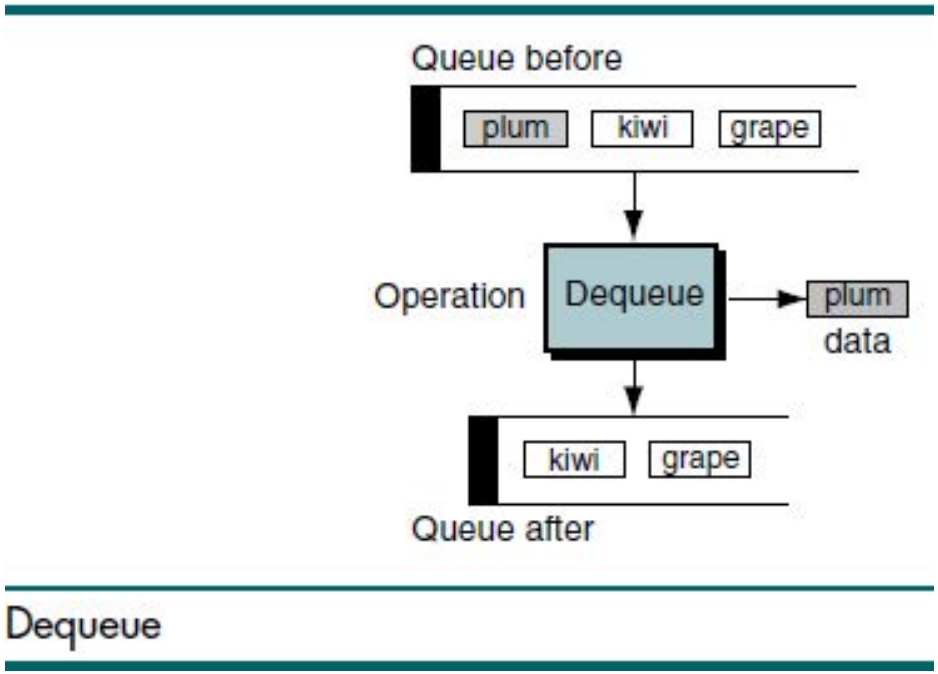
- A operação enfileirar é o único mecanismo de inserção de elementos em uma fila.
- Após a inserção de um elemento, esse elemento se torna o último da fila (*rear*).



Enqueue

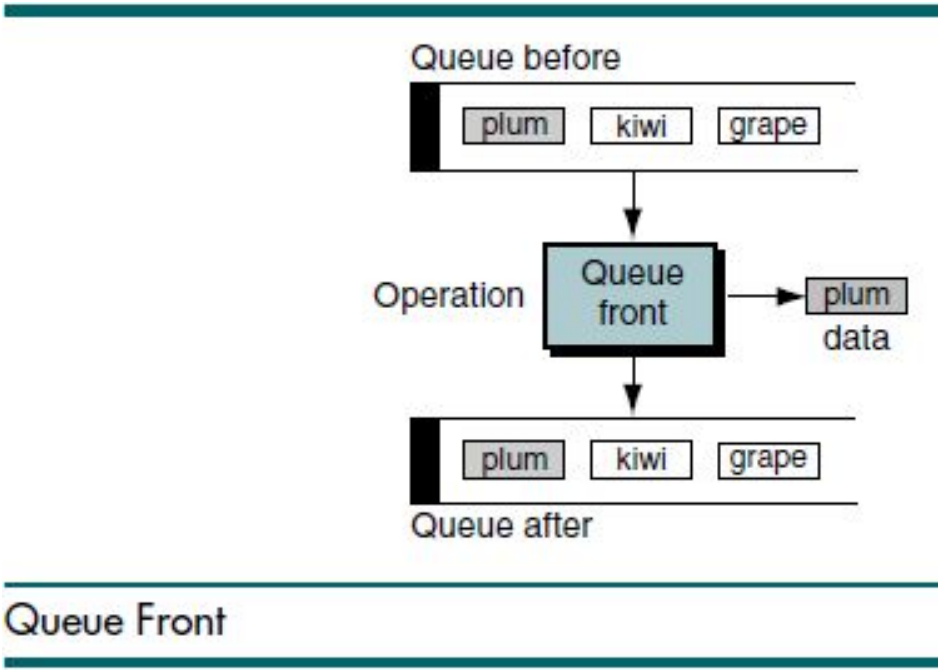
Filas (Queues)
Operação Desenfileirar (Dequeue)

- A operação desenfileirar é o único mecanismo de remoção de elementos em uma fila.
- O elemento no inicio da fila (front) é recuperado e removido da fila.



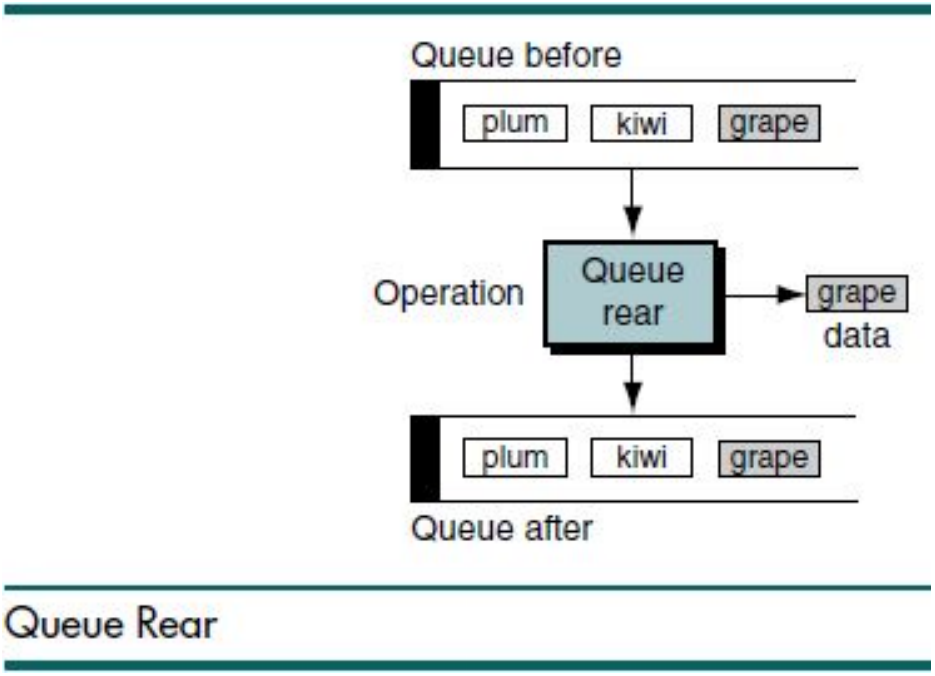
Filas (Queues)
Operação Inicio Fila (Queue Front)

- O elemento no inicio da fila (front) é recuperado, no entanto, o conteúdo da fila permanece inalterado.



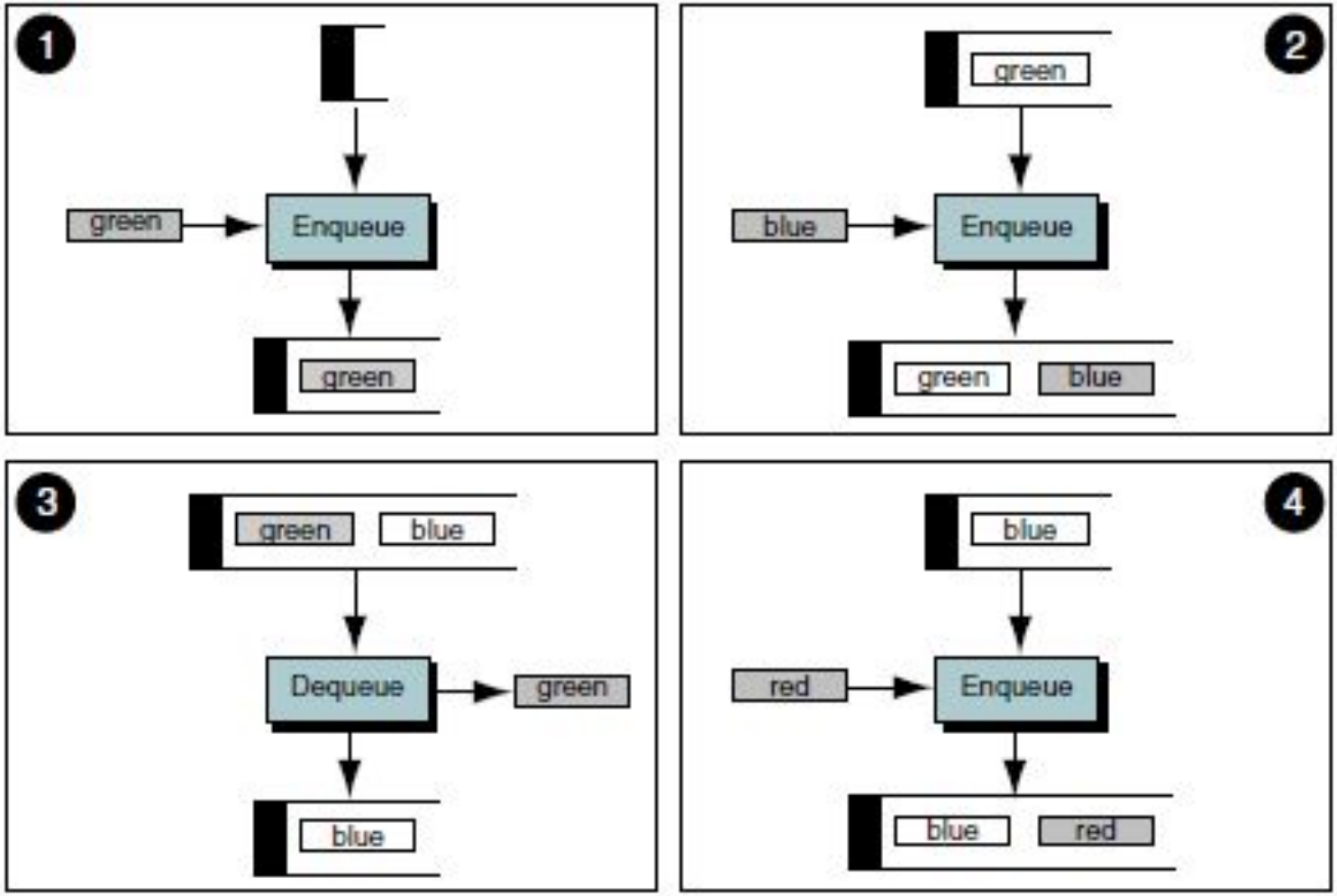
Filas (Queues)
Operação Fim Fila (Queue Rear)

- O elemento no fim da fila (rear) é recuperado, no entanto, o conteúdo da fila permanece inalterado.



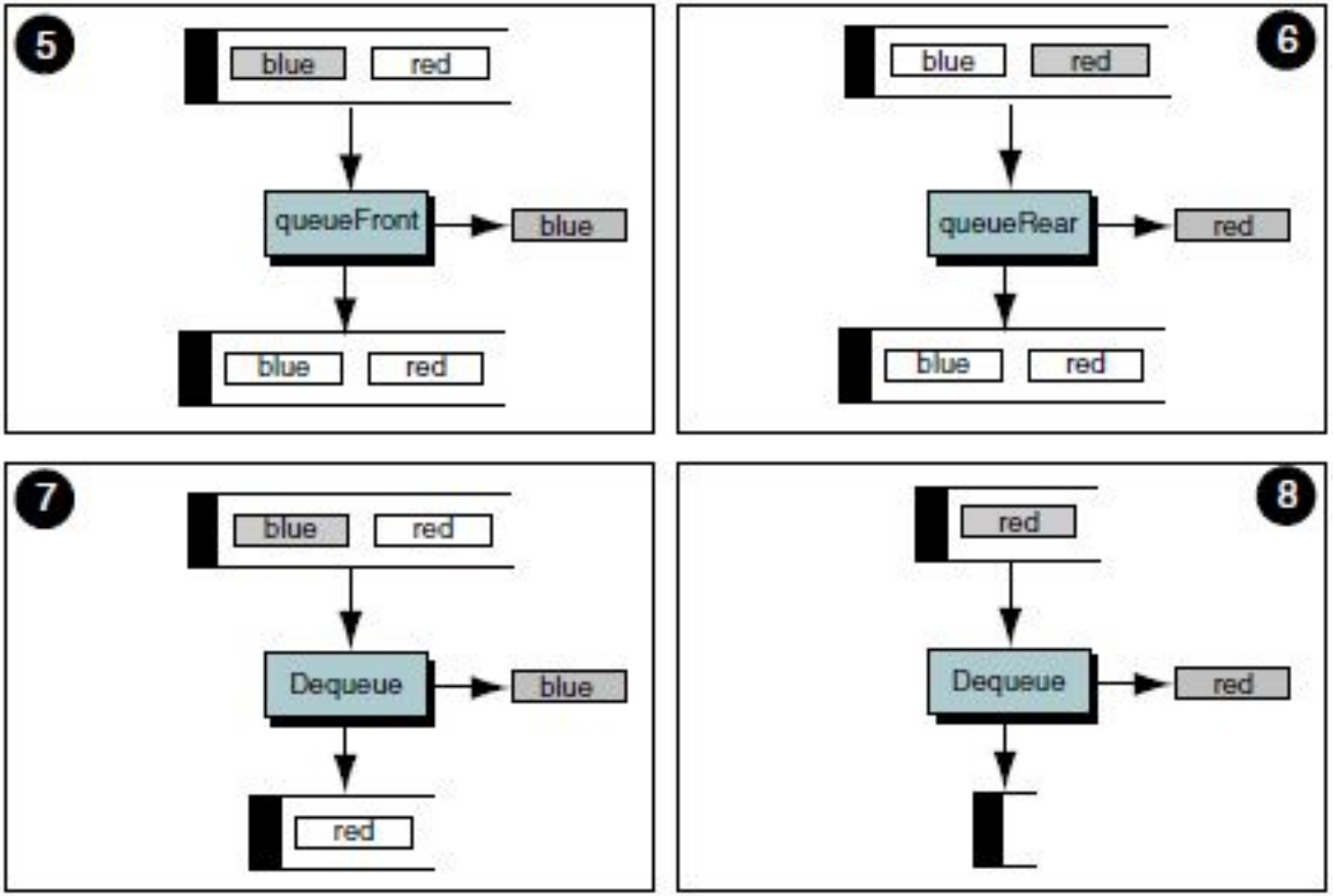
Filas (Queues)

Exemplo de Operação



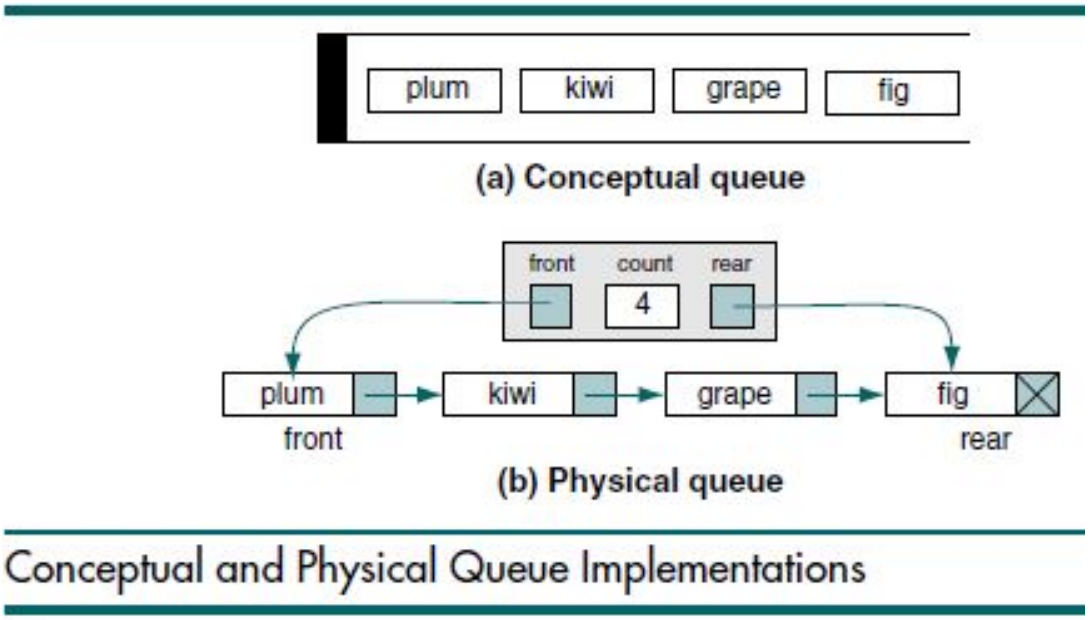
Filas (Queues)

Exemplo de Operação



Filas (Queues)
Implementação como Listas Encadeadas

- Estrutura conceitual e Estrutura Física como Lista Encadeada.



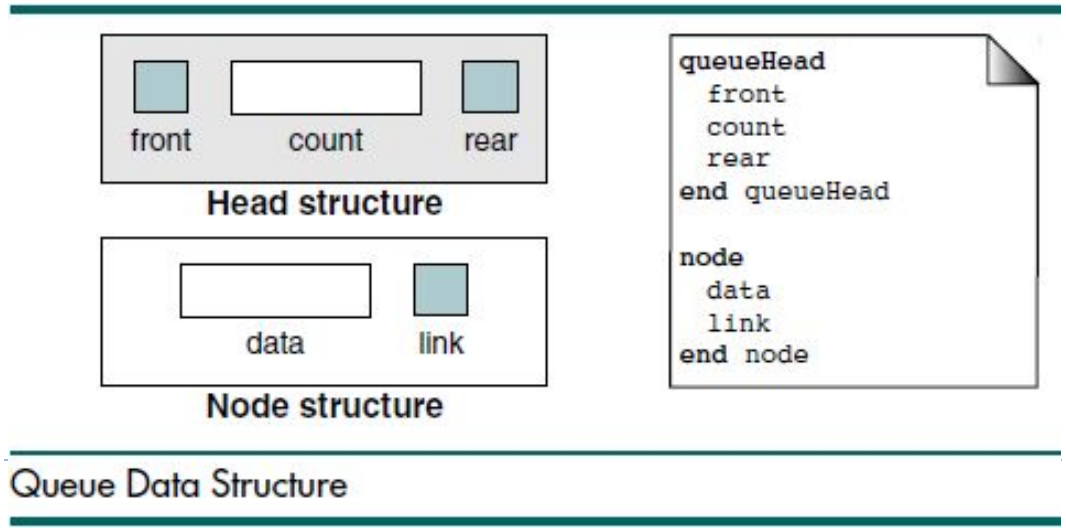
Conceptual and Physical Queue Implementations

- Precisamos de duas estruturas diferentes para implementar uma fila:
  - Uma estrutura de cabeçalho da fila (Queue Head);
  - Uma Estrutura de nó da fila (Queue node).

Filas (Queues)

Implementação como Listas Encadeadas - Estruturas

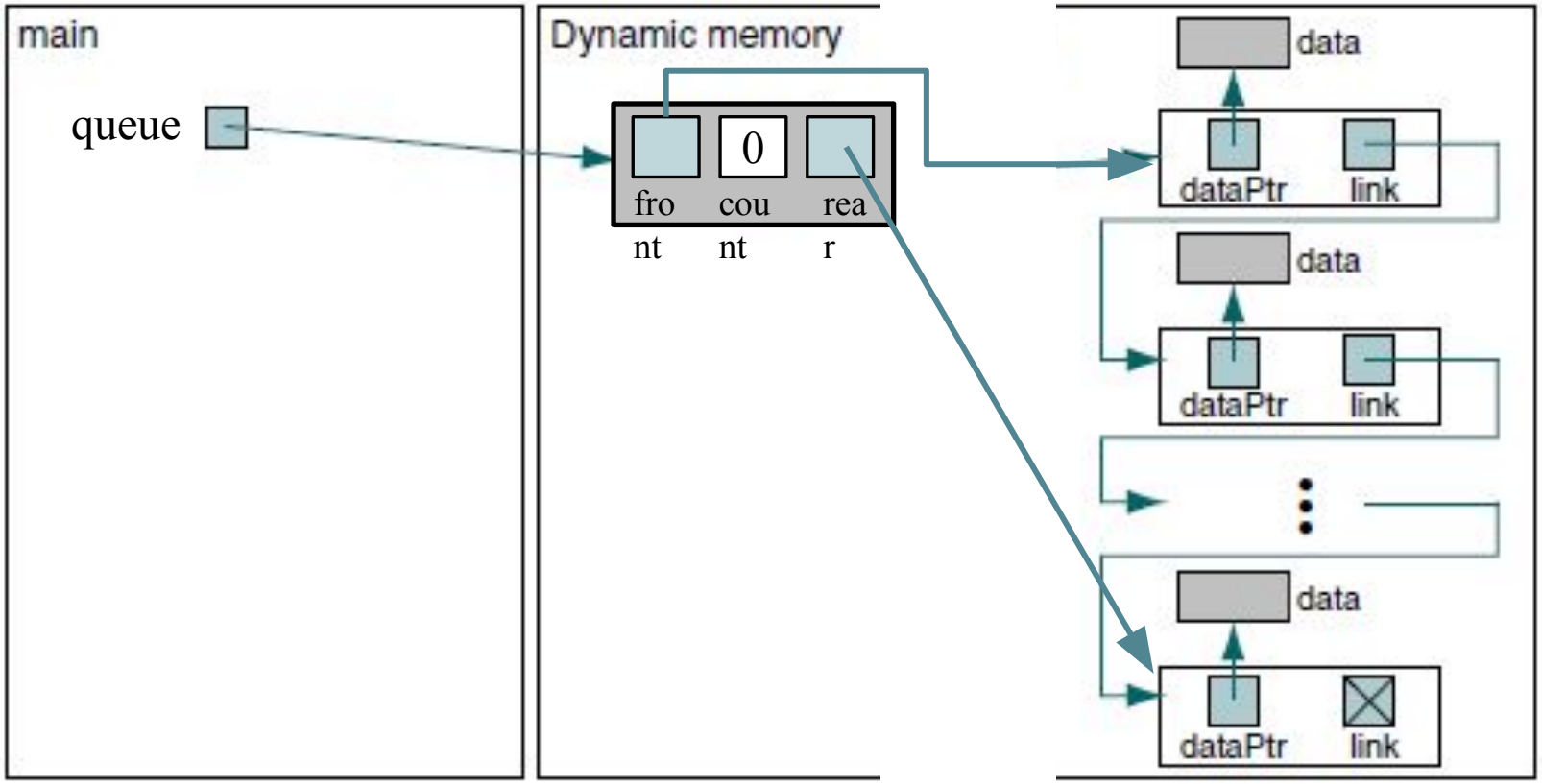
- **Cabeçalho da Fila (Queue Head).-** requer dois ponteiros e um contador de elementos. Um ponteiro aponta ao inicio da fila e o outro a fim da mesma. Outros atributos como o número total de elementos que foram processados pela fila podem ser guardados no cabeçalho.
- **Nó da fila (Queue node).-** contém os dados do usuário (ou um ponteiro para eles) e um ponteiro para outro nó da fila.
- Tanto os nós como o cabeçalho são armazenados em memória dinâmica.



Queue Data Structure

TAD Fila (Queue ADT)

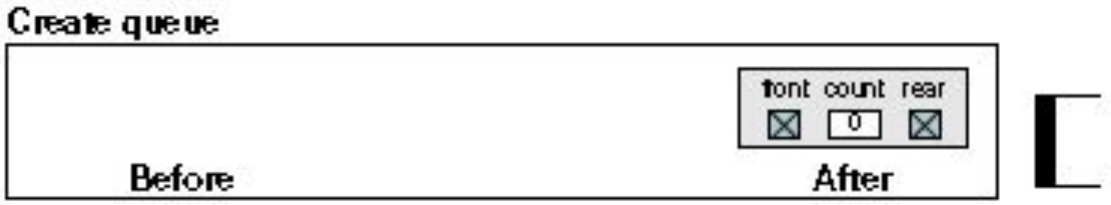
Implementação como Listas Encadeadas



Filas (Queues)

Implementação como Listas Encadeadas - Operações

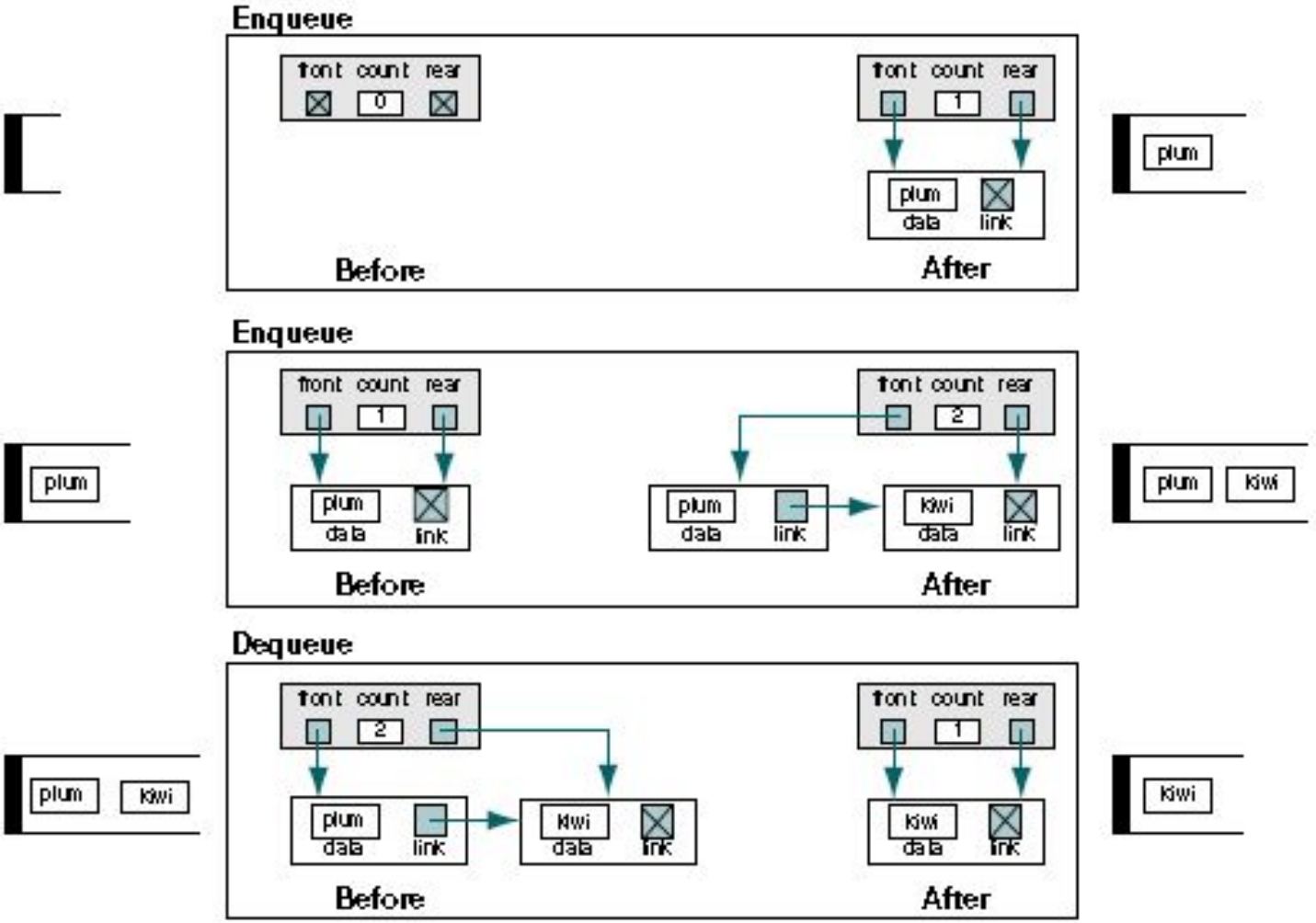
- As figuras ilustram as 4 operações de filas mais importantes:
  - Criar Fila (Create Queue)
  - Enfileirar (Inserir Fila) (Enqueue)
  - Desenfileirar (Remover Fila) (Dequeue)
  - Destruir Fila (Destroy Queue)



Filas (Queues)

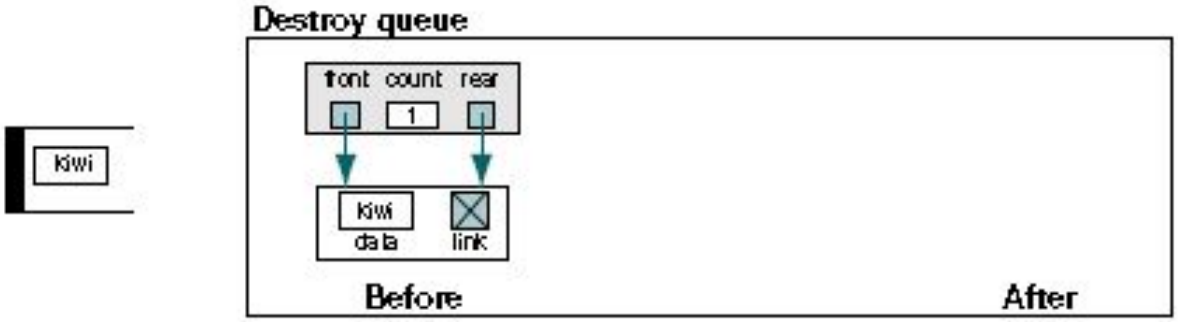
Implementação como Listas Encadeadas - Operações

- As figuras ilustram as 4 operações de filas mais importantes:



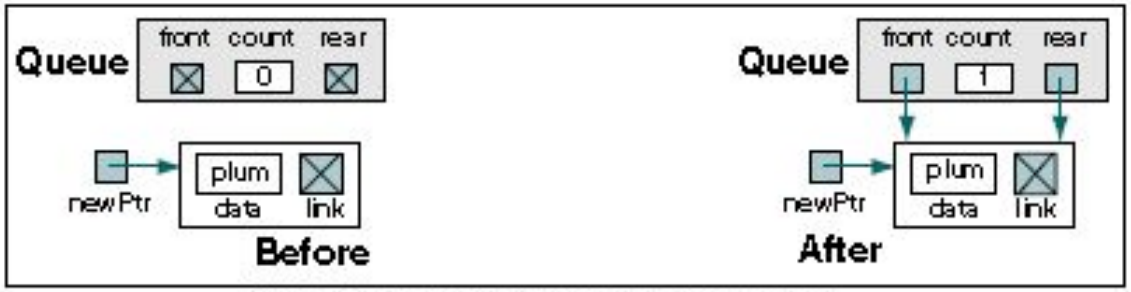
Filas (Queues)
Implementação como Listas Encadeadas - Operações

- As figuras ilustram as 4 operações de filas mais importantes:

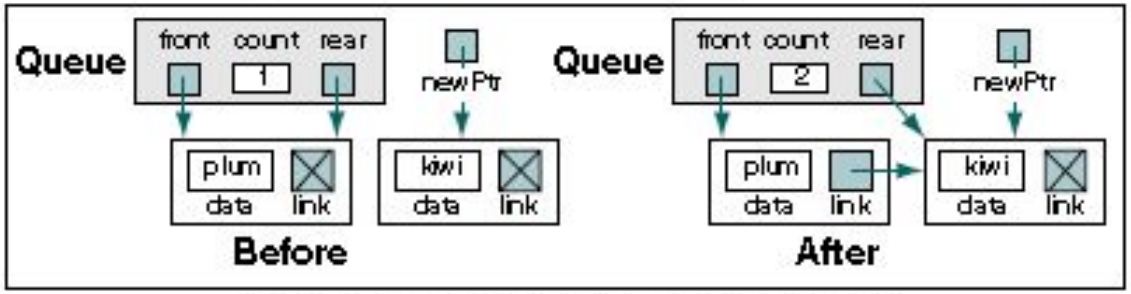


Filas (Queues)
Operação Enfileirar (Enqueue)

- Para inserir elementos em uma fila devemos considerar dois casos:
  - A fila está vazia;
  - A fila possui elementos;
- A figura ilustra os dois casos.



(a) Case 1: insert into empty queue



(b) Case 2: insert into queue with data

ponteiros front e rear apontam para o novo nó.

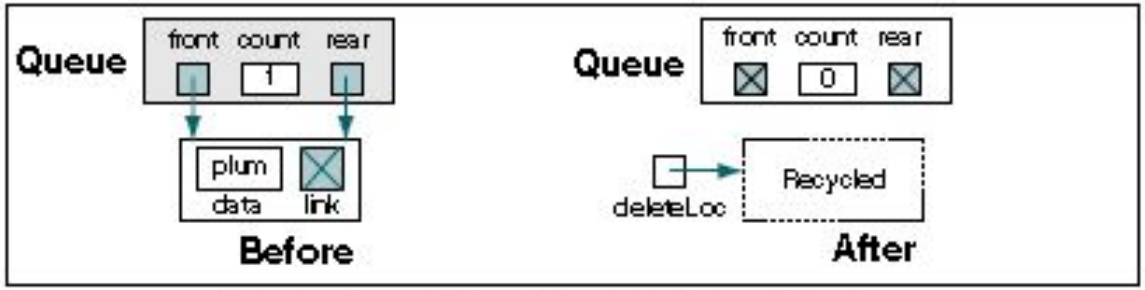
ponteiros link do ultimo nó e rear apontam para o novo nó.

Enqueue Example

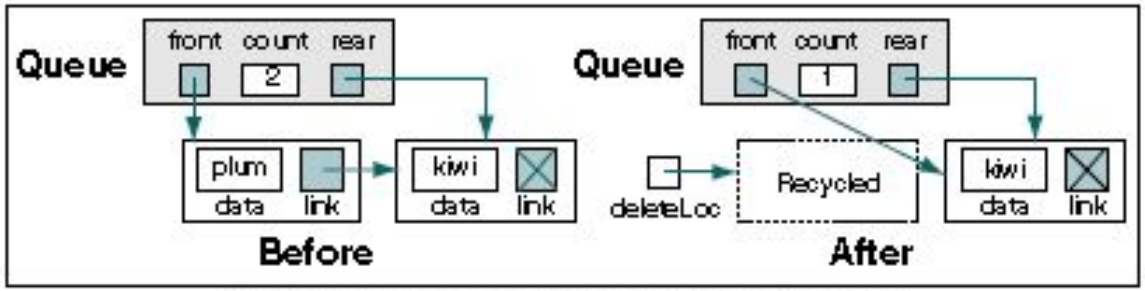
Filas (Queues)
Operação Desenfileirar (Dequeue)

- Para remover elementos em uma fila devemos considerar dois casos:
  - A fila possui um elemento;
  - A fila possui mais de um elemento;
- A figura ilustra os dois casos.

Fazer o ponteiro front igual ao ponteiro link do nó removido.



(a) Case 1: delete only item in queue



(b) Case 2: delete item at front of queue

Fazer rear null.

TAD Fila (Queue ADT)

Estrutura

- Em vez de armazenar um dado em cada nó, se armazena um ponteiro a esse dado.
- O programa de aplicação terá a responsabilidade de alocar memória para o dado e de passar o seu endereço ao TAD pilha.

TAD Fila (Queue ADT)

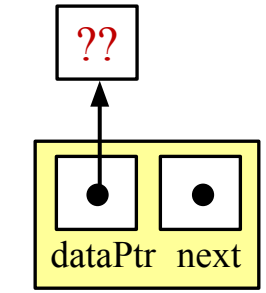
Estrutura da Fila

- As definições de tipos para a fila é mostrada no código:

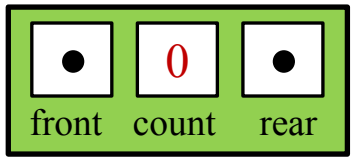
P4-01.h

```
1  //Queue ADT Type Defintions
2    typedef struct node
3      {
4       void*        dataPtr;
5       struct node* next;
6      } QUEUE_NODE;
7    typedef struct
8      {
9       QUEUE_NODE* front;
10      QUEUE_NODE* rear;
11      int         count;
12     } QUEUE;
13
```

```
// Definição do tipo nó da fila (QUEUE_NODE)

//  Ponteiro genérico
//  Ponteiro de ligação

//  Definição do tipo (cabeçalho) fila (QUEUE)

// Ponteiro ao primeiro nó da fila
 // Ponteiro ao último nó da fila
// Contador de elementos
```



QUEUE_NODE



QUEUE

TAD Fila (Queue ADT)

Estrutura da Fila

- Junto as definições de tipos devemos incluir os protótipos das funções.
- Estas definições devem ser incluídas em um arquivo cabeçalho (header file) de maneira que qualquer aplicação que precise definir uma fila possa fazê-lo facilmente.

P4-01.h (Continuação...)

```
14 //Prototype Declarations
15   QUEUE* createQueue  (void);
16   QUEUE* destroyQueue (QUEUE* queue);
17
18   bool  dequeue    (QUEUE* queue, void** itemPtr);
19   bool  enqueue    (QUEUE* queue, void*  itemPtr);
20   bool  queueFront (QUEUE* queue, void** itemPtr);
21   bool  queueRear  (QUEUE* queue, void** itemPtr);
22   int   queueCount (QUEUE* queue);
23
24   bool  emptyQueue (QUEUE* queue);
25   bool  fullQueue  (QUEUE* queue);
26 //End of Queue ADT Definitions
```

TAD Fila (Queue ADT)

Operações de suporte do TAD Fila

- Devemos ter operações de suporte ao funcionamento da Fila.
- Serão apresentadas as seguintes operações:
  - Criar Fila (Create Queue);
  - Enfileirar (Enqueue);
  - Desenfileirar (Dequeue);
  - Inicio Fila (Queue Front);
  - Fim Fila (Queue Rear);
  - Fila Vazia (Empty Queue);
  - Fila Cheia (Full Queue);
  - Contador Fila (Queue Count);
  - Destruir Fila (Destroy Queue).
- Essas operações serão implementadas em C.
- Obs. Todos os arquivos definindo o TAD Fila e as suas funções foram reunidos no arquivo queues.h.

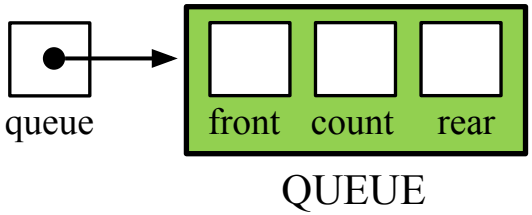
TAD Fila (Queue ADT)

Criar Fila (Create Queue)

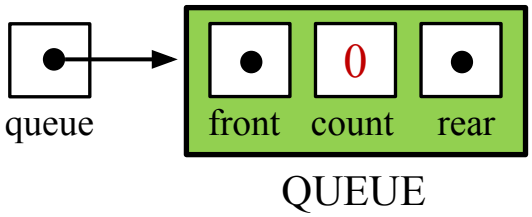
- Criar fila aloca memória para o nó cabeçalho da filha.
- Inicializa os ponteiros inicio e fim com nulo e fixa o contador em zero.
- Em caso de overflow, na tentativa de alocação de memória, a função retorna um ponteiro nulo.
- As figuras ilustram este processo.

1 QUEUE* queue;

2 queue:=(QUEUE*)malloc(sizeof(QUEUE));



3
```
queue->front:=NULL;
queue->rear:=NULL;
queue->count:=0;
```



TAD Fila (Queue ADT)

Criar Fila (Create Queue)

- O código para a função Criar Fila é o seguinte:
  P4-02.h

```
1  /*================ createQueue ================
2    Allocates memory for a queue head node from dynamic
3    memory and returns its address to the caller.
4       Pre    nothing
5       Post   head has been allocated and initialized
6       Return head if successful; null if overflow
7  */
8  QUEUE* createQueue (void)
9  {
10 //Local Definitions
11   QUEUE* queue;
12
13 //Statements
14   queue = (QUEUE*) malloc (sizeof (QUEUE));
15   if (queue)
16      {
17       queue->front  = NULL;
18       queue->rear   = NULL;
19       queue->count  = 0;
20      } // if
21   return queue;
22 } // createQueue
```

Cria uma fila vazia.

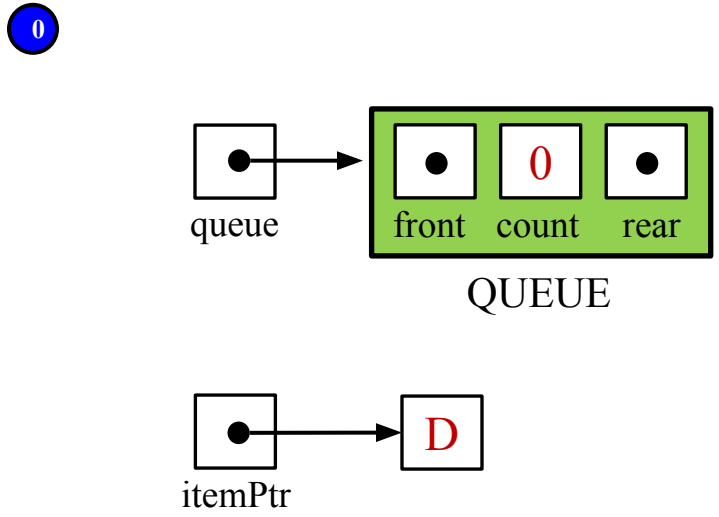
Entrada: Nenhuma.
Saída: Retorna um ponteiro a um cabeçalho (fila) ou um ponteiro nulo, caso não houver memória

```
// Retorna um ponteiro a (cabeçalho) pilha


// Ponteiro à (cabeçalho) fila

// Alocação de memória (cabeçalho) fila


// Ponteiro ao inicio é nulo
// Ponteiro ao fim é nulo
// Contador  é zerado

// Retorna ponteiro a (cabeçalho) fila
```

TAD Fila (Queue ADT)

Enfileirar (Enqueue)

- A operação Enfileirar aloca memória para um novo nó de dados.
- Se tiver sucesso na alocação, insere o novo nó no fim da fila (rear) e retorna true. Caso contrário, retorna false.
- As figuras ilustram este processo.

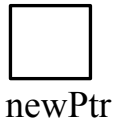


TAD Fila (Queue ADT)

Enfileirar (Enqueue)

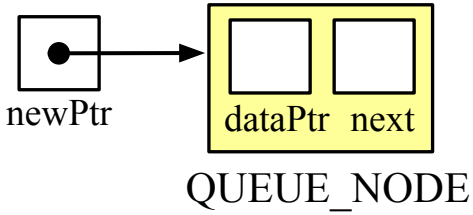
1

```
QUEUE_NODE*   newPtr;
```



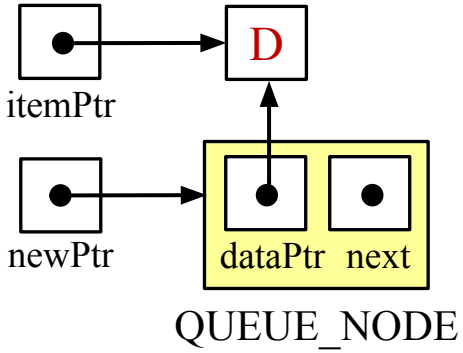
2

```
newPtr:=(QUEUE_NODE*)malloc(sizeof(QUEUE_NODE));
```



3

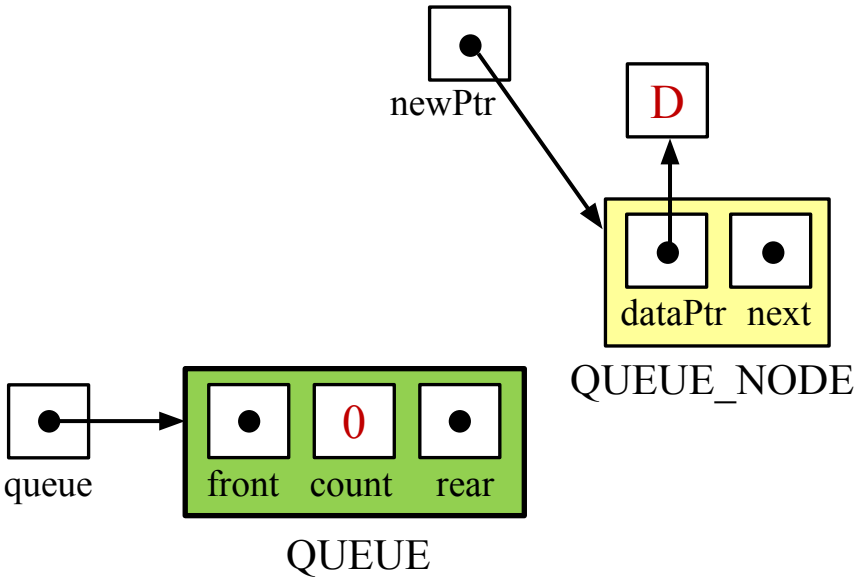
```
newPtr->dataPtr:=itemPtr;
newPtr->next:=NULL;
```



TAD Fila (Queue ADT)

Enfileirar (Enqueue)

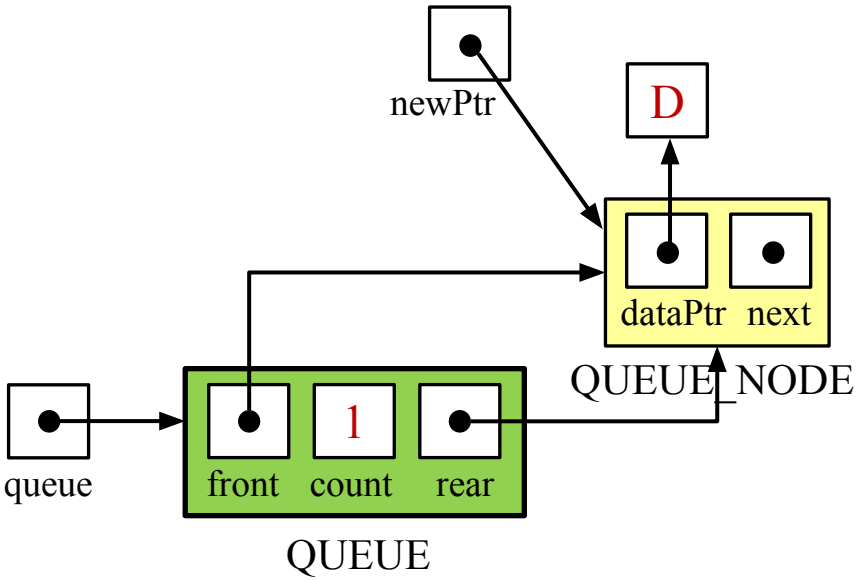
```
If (queue->count==0)
    queue->front :=newPtr;
    queue->rear:=newPtr;
    (queue->count)++;
```



TAD Fila (Queue ADT)

Enfileirar (Enqueue)

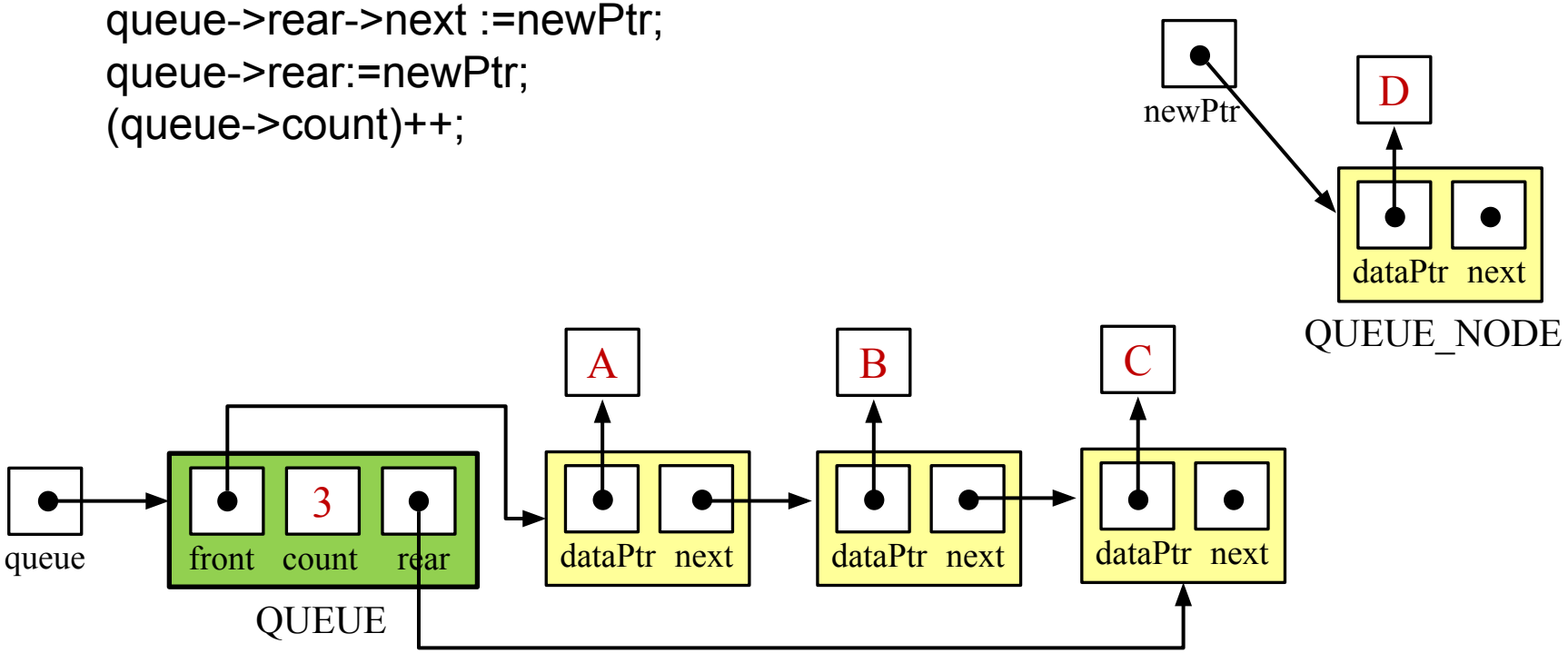
```
If (queue->count==0)
    queue->front :=newPtr;
    queue->rear:=newPtr;
    (queue->count)++;
```



TAD Fila (Queue ADT)

Enfileirar (Enqueue)

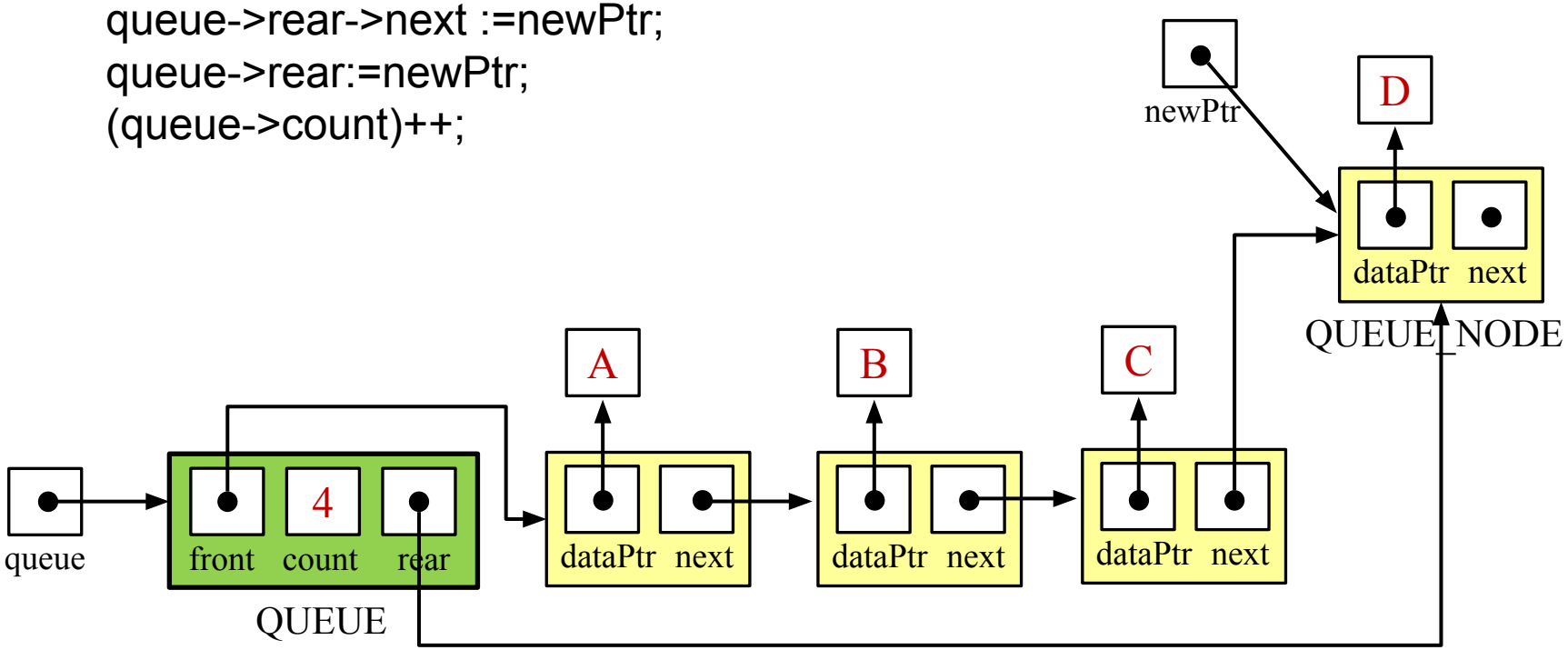
```
If (queue->count!=0)
    queue->rear->next :=newPtr;
    queue->rear:=newPtr;
    (queue->count)++;
```



TAD Fila (Queue ADT)

Enfileirar (Enqueue)

```
If (queue->count!=0)
    queue->rear->next :=newPtr;
    queue->rear:=newPtr;
    (queue->count)++;
```



TAD Fila (Queue ADT)

P4-03.h

Enfileirar (Enqueue)

```
1  /*================= enqueue ================
2    This algorithm inserts data into a queue.
3       Pre    queue has been created
4       Post   data have been inserted
5       Return true if successful, false if overflow
6  */
7  bool enqueue (QUEUE* queue, void* itemPtr)
8  {
9  //Local Definitions
10   QUEUE_NODE* newPtr;
11
12 //Statements
13   if (!(newPtr =
14      (QUEUE_NODE*)malloc(sizeof(QUEUE_NODE))))
15      return false;
16
17   newPtr->dataPtr = itemPtr;
18   newPtr->next    = NULL;
19
20   if (queue->count == 0)
21      // Inserting into null queue
22      queue->front  = newPtr;
23   else
24      queue->rear->next = newPtr;
25
26   (queue->count)++;
27   queue->rear = newPtr;
28   return true;
29 } // enqueue
```

TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- A operação Desenfileirar começa verificando se existem dados na fila.
- Caso existam, retorna o endereço do dado no inicio da fila (front). Redefine os ponteiros de maneira a eliminar o nó que ocupa a primeira posição da fila. Subtrai um do contador de nós. Retorna true.
- Caso a fila esteja vazia, retorna false.

- O código desta operação é mostrado a seguir:
- As figuras ilustram este processo.

TAD Fila (Queue ADT)

Desenfileirar (Dequeue)

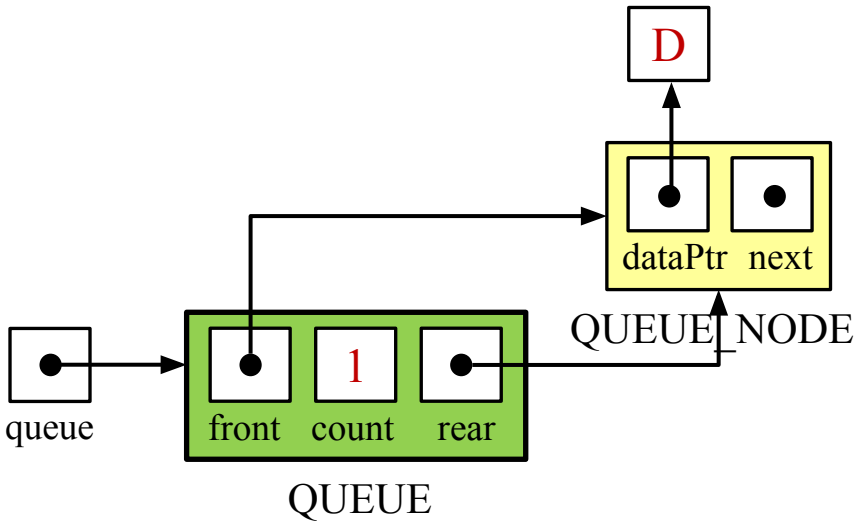
P4-04.h

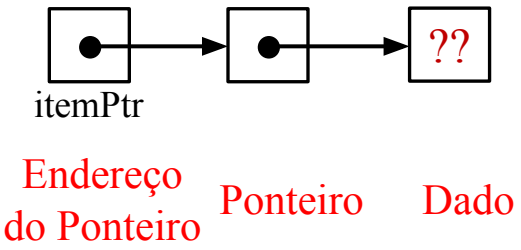
```
1  /*================= dequeue ================
2    This algorithm deletes a node from the queue.
3       Pre    queue has been created
4       Post   Data pointer to queue front returned and
5              front element deleted and recycled.
6       Return true if successful; false if underflow
7  */
8  bool dequeue (QUEUE* queue, void** itemPtr)
9  {
10 //Local Definitions
11   QUEUE_NODE* deleteLoc;
12
13 //Statements
14   if (!queue->count)
15       return false;
16
17   *itemPtr  = queue->front->dataPtr;
18   deleteLoc = queue->front;
19   if (queue->count == 1)
20      // Deleting only item in queue
21      queue->rear  = queue->front = NULL;
22   else
23      queue->front = queue->front->next;
24   (queue->count)--;
25   free (deleteLoc);
26
27   return true;
28 } // dequeue
```

TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Inicialmente temos:

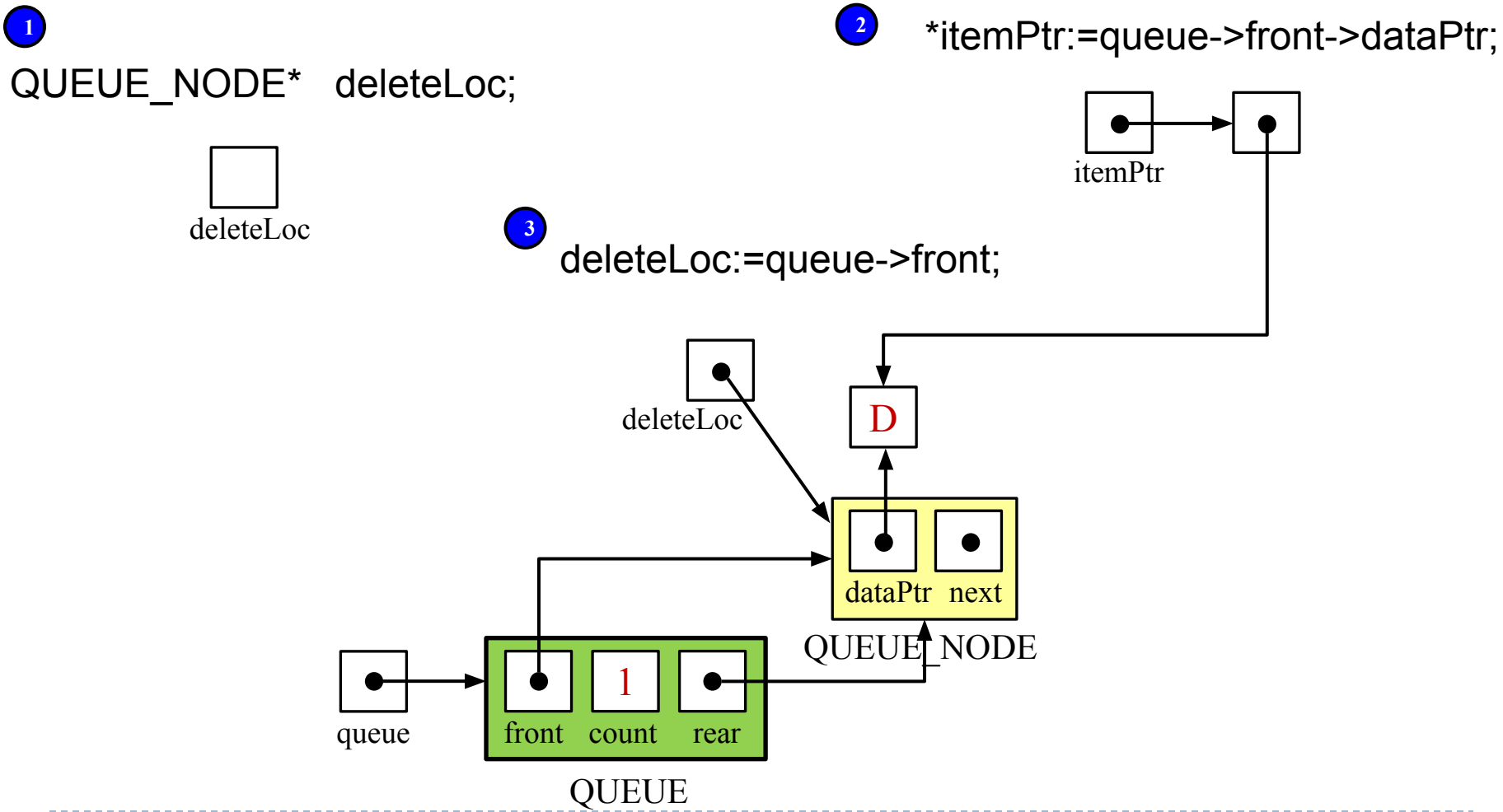
```
bool  dequeue  (QUEUE*  queue,  void**  itemPtr)
```





TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Temos dois casos possíveis. Se a fila tem apenas um elemento:

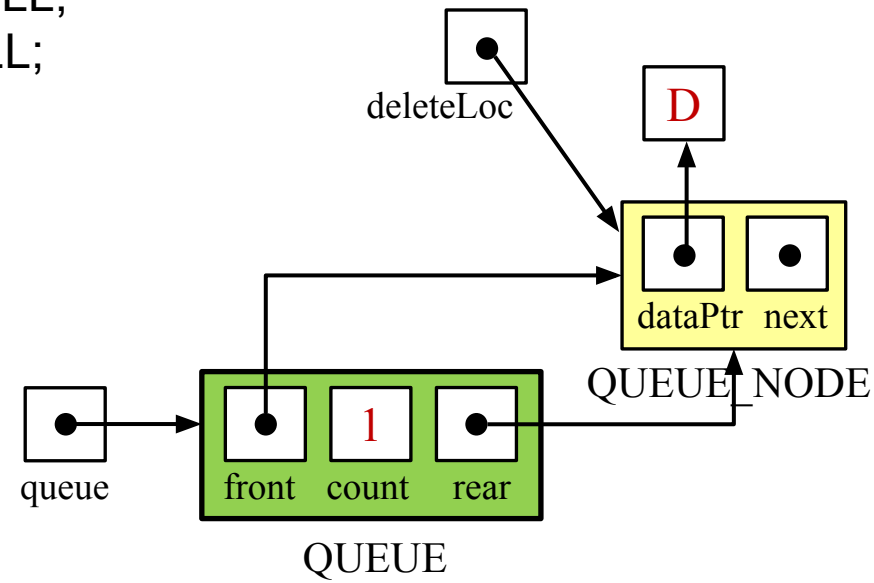


TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Se a fila tem apenas um elemento:

4

```
If (queue->count==1)
    queue->front :=NULL;
    queue->rear:=NULL;
    (queue->count)--;
    free(deleteLoc);
    return true;
```

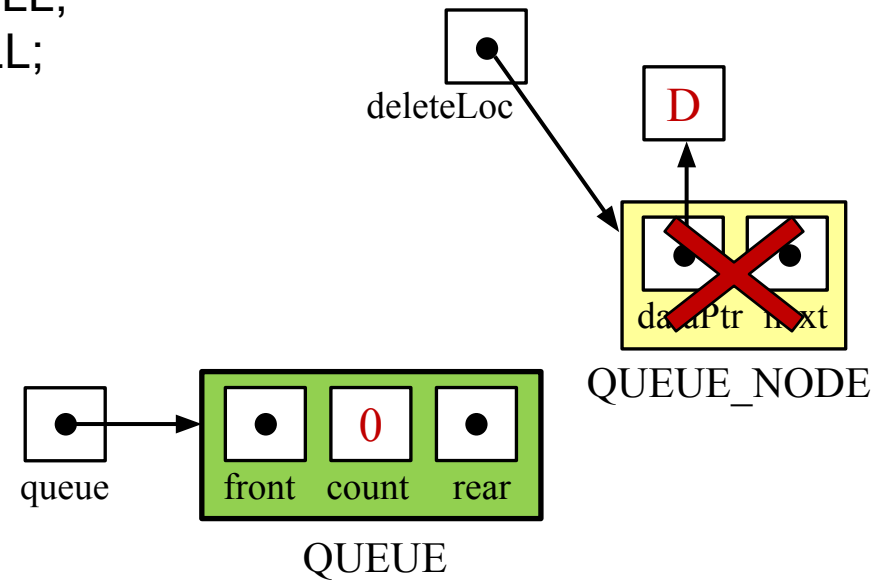


TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Se a fila tem apenas um elemento:

4

```
If (queue->count==1)
    queue->front :=NULL;
    queue->rear:=NULL;
    (queue->count)--;
    free(deleteLoc);
    return true;
```



TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

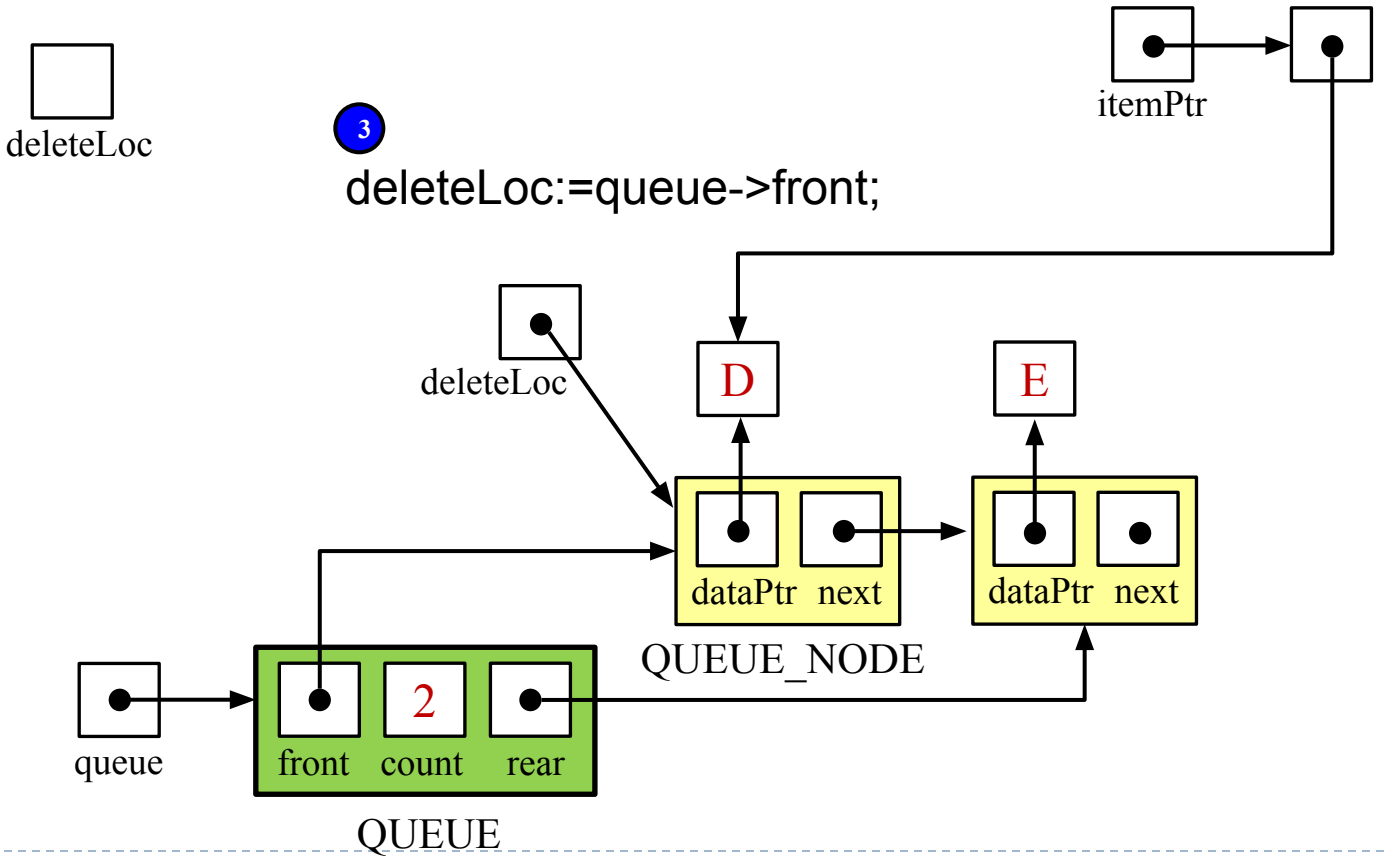
- Temos dois casos possíveis. Se a fila mais de um elemento:

1

```
QUEUE_NODE*   deleteLoc;
```

2

```
*itemPtr:=queue->front->dataPtr;
```

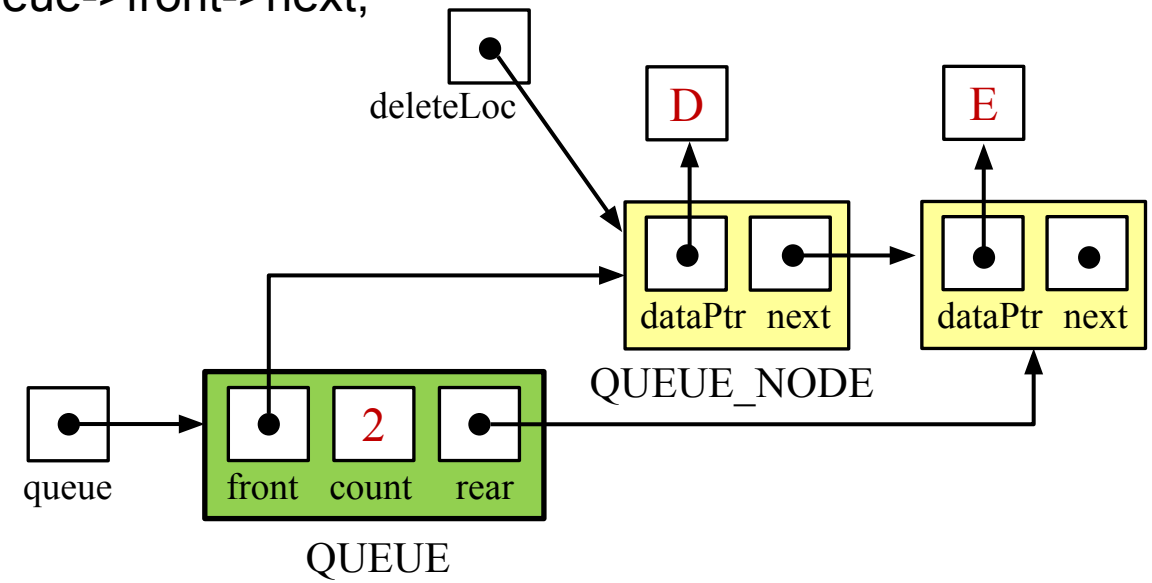


TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Se a fila mais de um elemento:

4

```
If (queue->count>1)
    queue->front := queue->front->next;
    (queue->count)--;
    free(deleteLoc);
    return true;
```

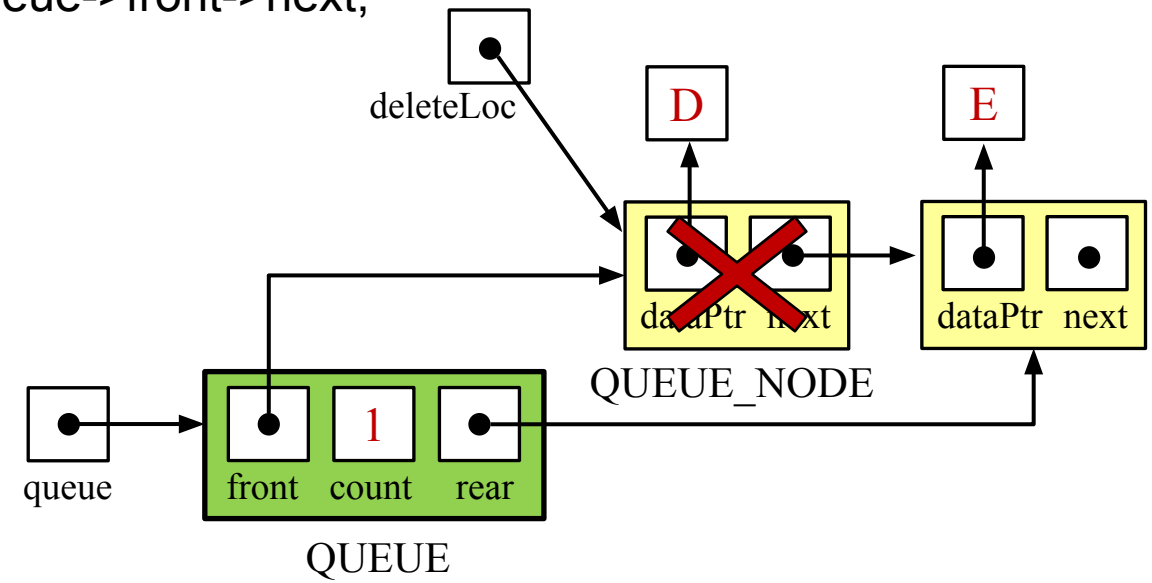


TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Se a fila mais de um elemento:

4

```
If (queue->count>1)
    queue->front := queue->front->next;
    (queue->count)--;
    free(deleteLoc);
    return true;
```

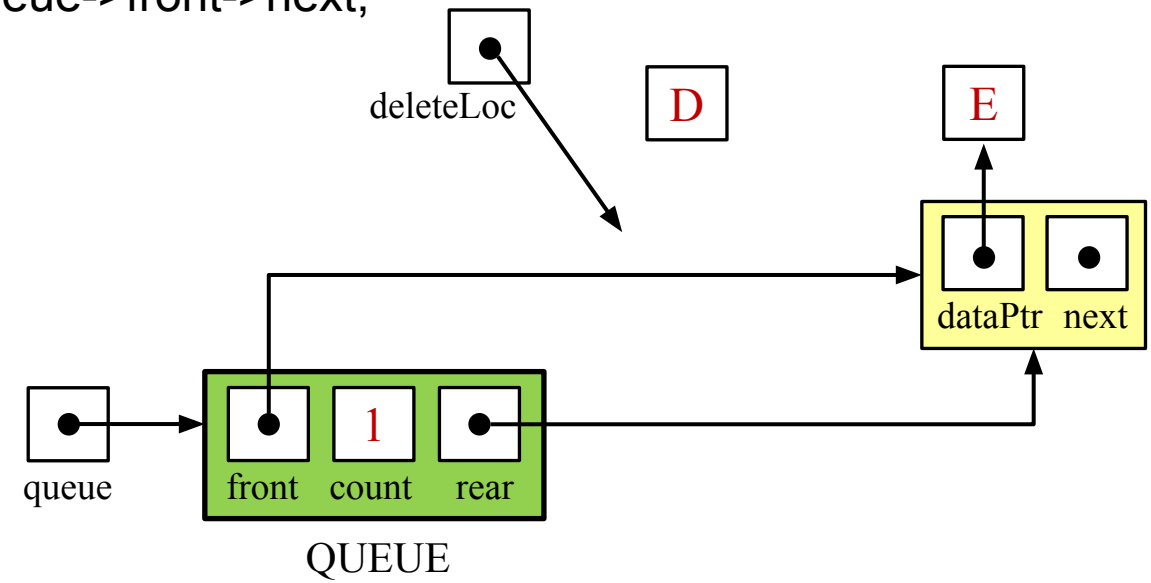


TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Se a fila mais de um elemento:

4

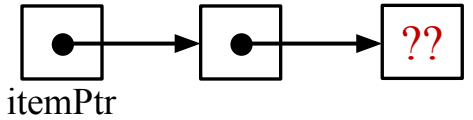
```
If (queue->count>1)
    queue->front := queue->front->next;
    (queue->count)--;
    free(deleteLoc);
    return true;
```



TAD Fila (Queue ADT)
Desenfileirar (Dequeue)

- Considerando a definição da função desenfileirar (dequeue):

```
bool    dequeue    (QUEUE*  queue,   void**  itemPtr)
```



- A chamada essa função contém vários detalhes:

```
QUEUE*  fila1;
int*        pdado;

dequeue    (fila1, (void*) &pdado)
```

- Como a função retorna um booleano a chamada dessa função deve ocorrer em uma sentença condicional:

```
If dequeue    (fila1, (void*) &pdado)
```

TAD Fila (Queue ADT)
Inicio Fila (Queue Front)

- A operação Inicio Fila (Queue Front) repassa o endereço do dado no inicio da fila a função de chamada. Caso a fila tenha um ou mais elementos retorna true. Caso contrário, retorna false.
- Difere da operação desenfileirar no fato de que a fila não é modificada.

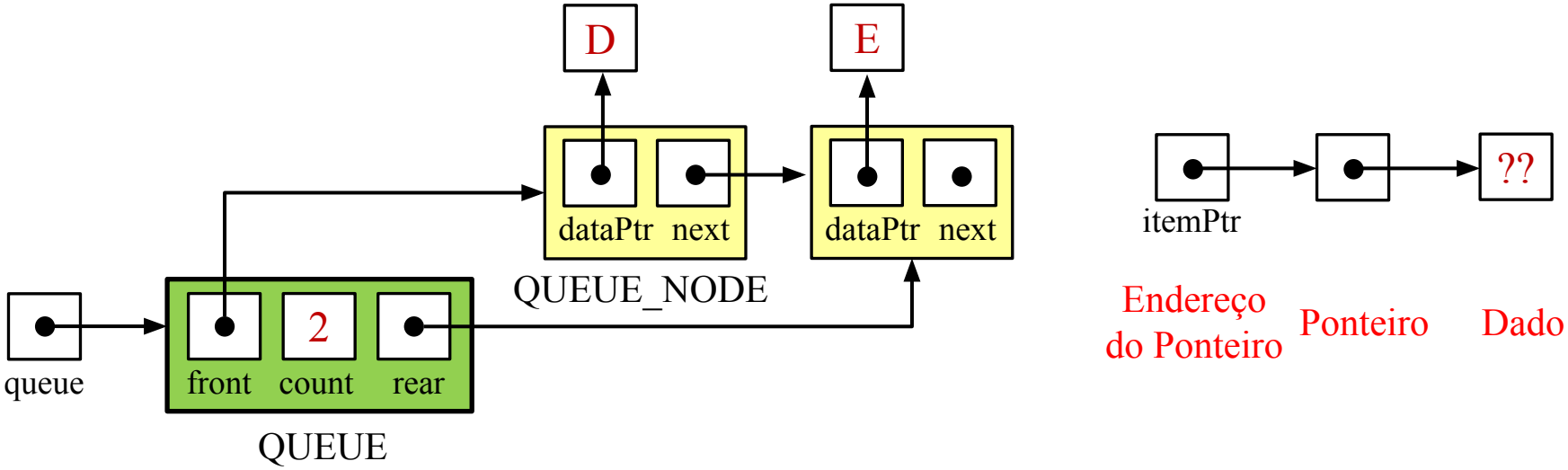
P4-05.h

```
1  /*================== queueFront ================
2    This algorithm retrieves data at front of the
3    queue without changing the queue contents.
4       Pre    queue is pointer to an initialized queue
5       Post   itemPtr passed back to caller
6       Return true if successful; false if underflow
7  */
8  bool queueFront (QUEUE* queue, void** itemPtr)
9  {
10 //Statements
11   if (!queue->count)
12       return false;
13   else
14      {
15       *itemPtr = queue->front->dataPtr;
16        return true;
17      } // else
18 } // queueFront
```

TAD Fila (Queue ADT)
Inicio Fila (Queue Front)

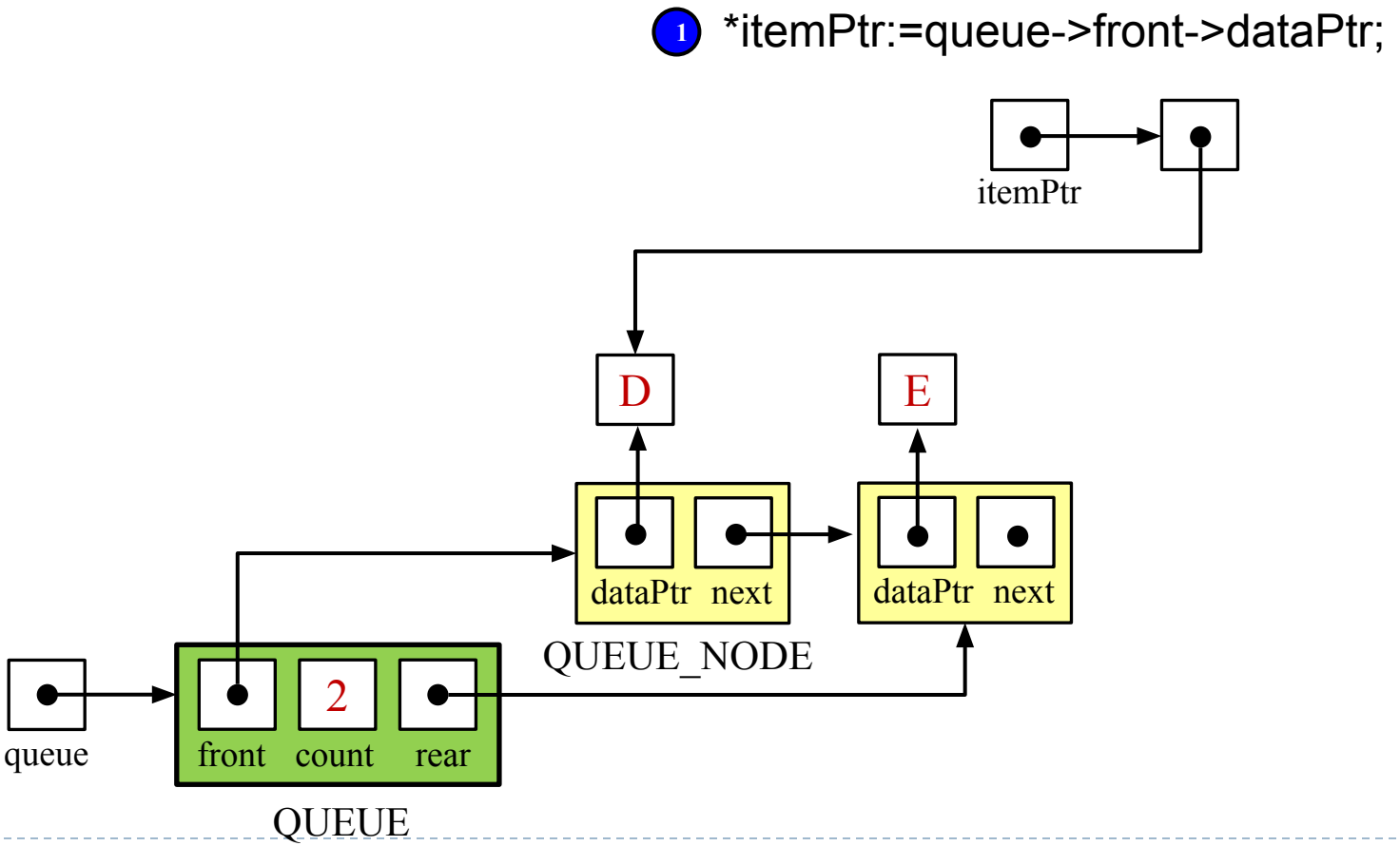
- Os parâmetros da função Inicio Fila (Queue Front) são os seguintes:

```
bool  queueFront  (QUEUE*  queue,  void**  itemPtr)
```



TAD Fila (Queue ADT)
Inicio Fila (Queue Front)

- A função recupera o ponteiro ao dado associado ao inicio da Fila.
- Copia esse ponteiro e repassa à função de chamada.



TAD Fila (Queue ADT)
Fim Fila (Queue Rear)

- A função Fim Fila (Queue Rear) é quase idêntica a função Inicio Fila (Queue Front), com a diferença de que repassa o endereço do dado no fim da fila a função de chamada. Caso a fila tenha um ou mais elementos retorna true. Caso contrário, retorna false. A fila não é modificada.

P4-06.h

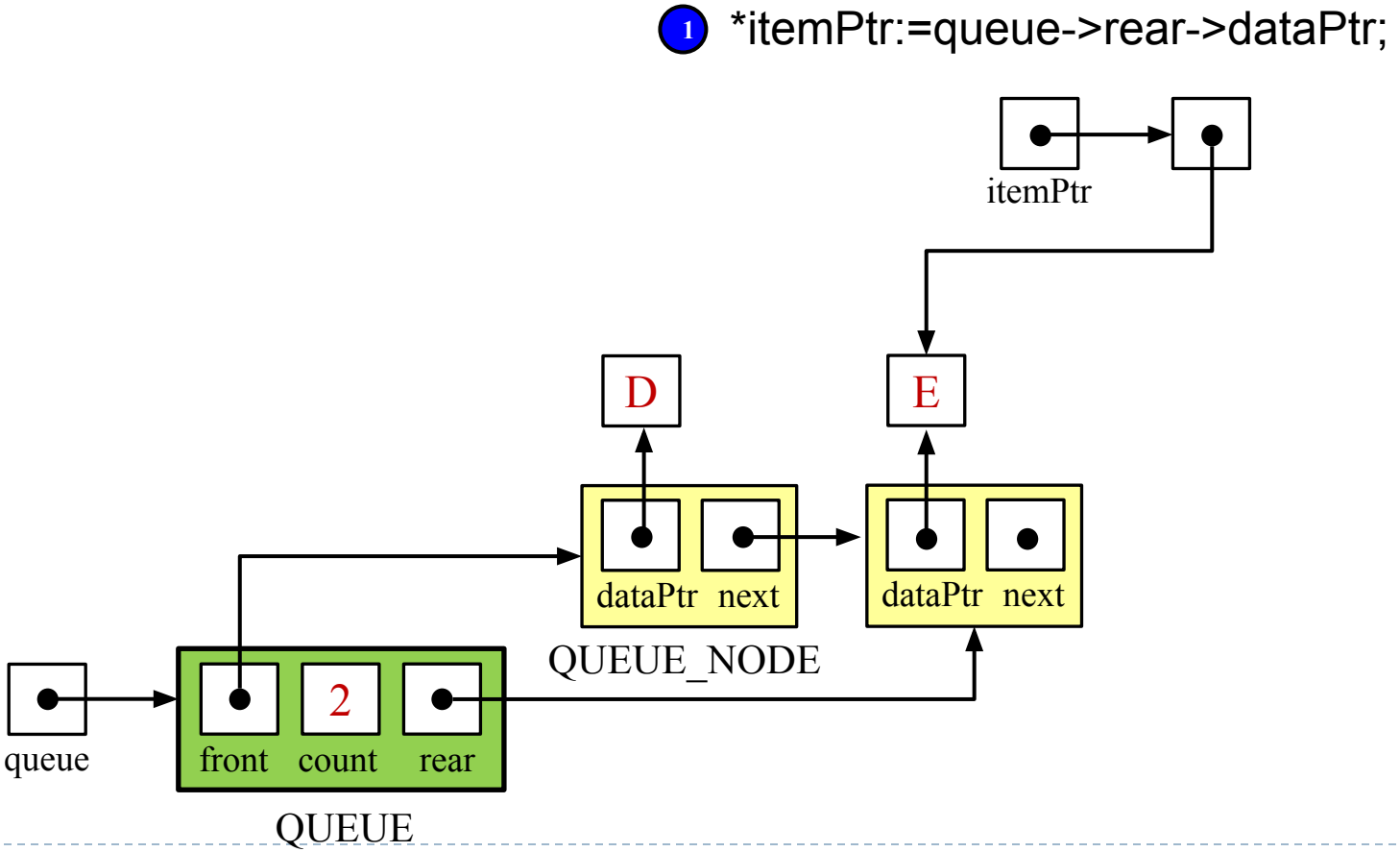
```
1  /*================== queueRear ================
2    Retrieves data at the rear of the queue
3    without changing the queue contents.
4       Pre    queue is pointer to initialized queue
5       Post   Data passed back to caller
6       Return true if successful; false if underflow
7  */
8  bool queueRear (QUEUE* queue, void** itemPtr)
9  {
10 //Statements
11    if (!queue->count)
12       return true;
13    else
14       {
15        *itemPtr = queue->rear->dataPtr;
16        return false;
17       } // else
18 } // queueRear
```

false;

true;

TAD Fila (Queue ADT)
Fim Fila (Queue Rear)

- A função recupera o ponteiro ao dado associado ao fim da Fila.
- Copia esse ponteiro e repassa à função de chamada.



TAD Fila (Queue ADT)

Fila Vazia (Empty Queue)

- A função Fila Vazia (Empty Queue) verifica se a fila está vazia.
- Se o contador do número de elementos está zerado, retorna true.
- Caso contrário, retorna false.

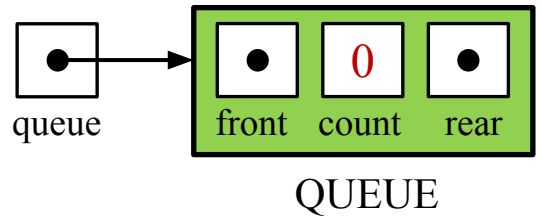
P4-07.h

```
1  /*================== emptyQueue ================
2    This algorithm checks to see if queue is empty.
3    Pre    queue is a pointer to a queue head node
4    Return true if empty; false if queue has data
5  */
6  bool emptyQueue (QUEUE* queue)
7  {
8  //Statements
9    return (queue->count == 0);
10 } // emptyQueue
```

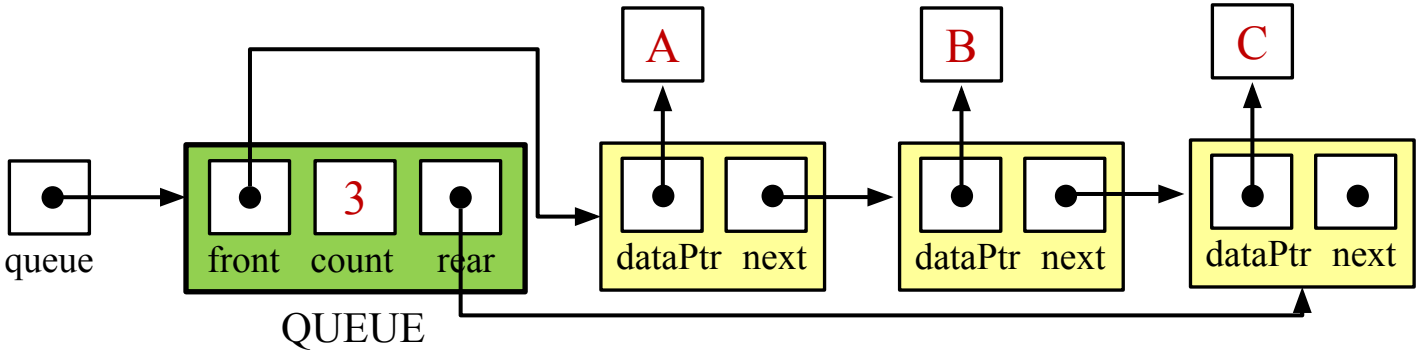
TAD Fila (Queue ADT)

Fila Vazia (Empty Queue)

- A figura ilustra os dois casos possíveis para a função Fila Vazia (Empty Queue).



Retorna True!!!



Retorna False!!!

TAD Fila (Queue ADT)

Fila Cheia (Full Queue)

- A função Fila Cheia (Full Queue) verifica se a fila está cheia.
- A linguagem C não fornece uma maneira de testar se existe espaço disponível na memória dinâmica.
- Uma forma de fazer isso é realizando uma tentativa de alocação de um nó e verificar se a tentativa foi bem sucedida ou não.
- Se o nó for alocado, esse nó deve ser liberado e retorna-se false. A fila não está cheia.
- Caso contrário, retorna-se true. A fila está cheia.

TAD Fila (Queue ADT)
Fila Cheia (Full Queue)

- O código da função Fila Cheia (Full Queue) é mostrado a seguir:

P4-08.h

```
1  /*================== fullQueue ================
2    This algorithm checks to see if queue is full. It
3    is full if memory cannot be allocated for next node.
4       Pre    queue is a pointer to a queue head node
5       Return true if full; false if room for a node
6  */
7  bool fullQueue (QUEUE* queue)
8  {
9  //Local Definitions
10 QUEUE_NODE* temp;
11
12 //Statements
13   temp = (QUEUE_NODE*)malloc(sizeof(*(queue->rear)));
14   if (temp)
15       {
16        free (temp);
17        return  false;
18       } // if
19   // Heap full
20   return   true;
21 } // fullQueue
```

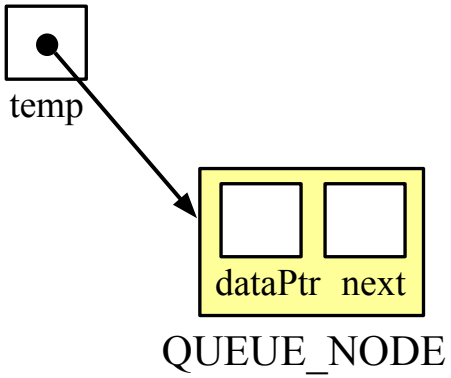
TAD Fila (Queue ADT)
Fila Cheia (Full Queue)

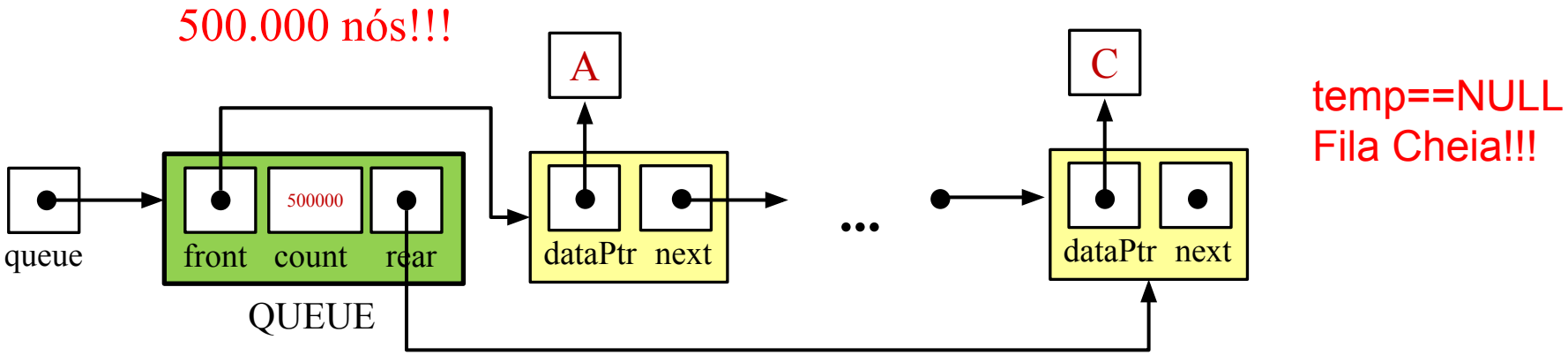
- A figura ilustra a função Fila Cheia (Full Queue):

temp:=(QUEUE_NODE*)malloc(sizeof(QUEUE_NODE));

ou

temp:=(QUEUE_NODE*)malloc(sizeof(*(queue->rear)));





TAD Fila (Queue ADT)
Contador Fila (Queue Count)

- A função Contador Fila (Queue Count) simplesmente retorna o valor do contador que se encontra no nó cabeça-lho da fila.
- O código da função é o seguinte:

P4-09.h

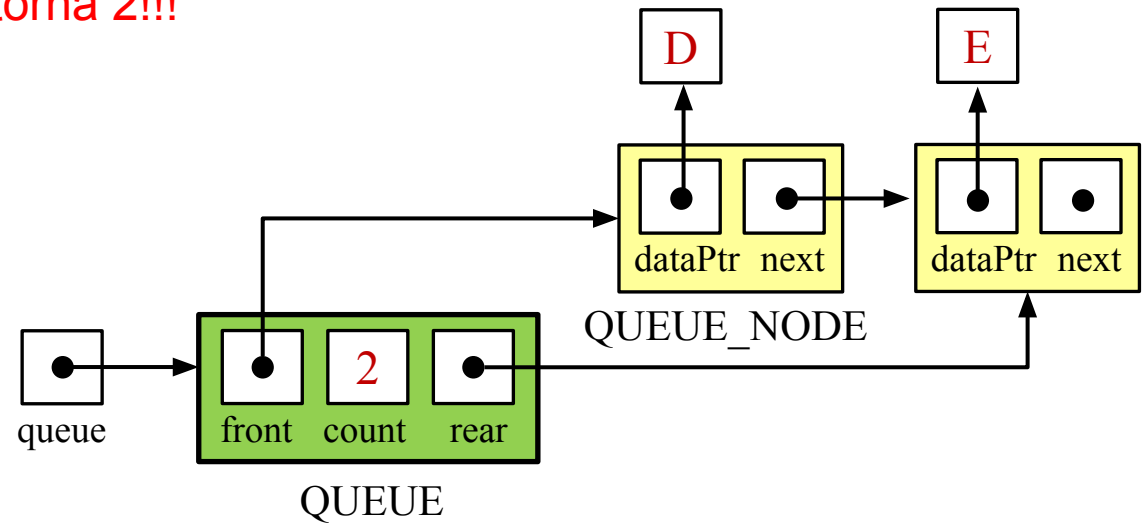
```
1  /*================== queueCount ================
2    Returns the number of elements in the queue.
3       Pre    queue is pointer to the queue head node
4       Return queue count
5  */
6  int queueCount(QUEUE* queue)
7  {
8  //Statements
9    return queue->count;
10 } // queueCount
```

TAD Fila (Queue ADT)
Contador Fila (Queue Count)

- A figura ilustra a função Contador Fila (Queue Count).

```
return   queue->count;
```

Retorna 2!!!



TAD Fila (Queue ADT)

Destruir Fila (Destroy Queue)

- A função Destruir Fila (Destroy Count) considera dois casos.
- O caso mais simples acontece se a fila está vazia. Neste caso, libera-se a memória alocada ao cabeçalho da fila e retorna-se um ponteiro nulo.
- O outro caso acontece se a fila contém elementos. Neste caso, precisamos liberar tanto a memória alocada aos dados de cada nó quanto a memória alocada a estrutura de cada nó.
- Começando pelo primeiro nó, liberamos primeiro o dado e depois a estrutura.
- Após processar todos os nós, libera-se a memória alocada ao cabeçalho da fila e retorna-se um ponteiro nulo.

TAD Fila (Queue ADT)

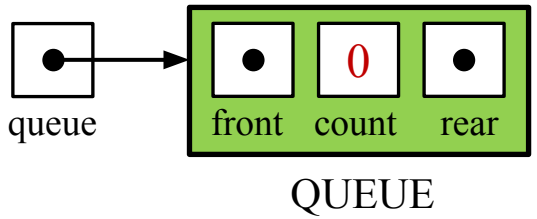
Destruir Fila (Destroy Queue)

P4-10.h

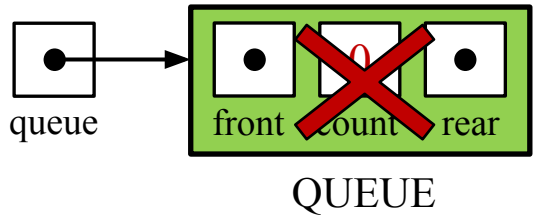
```
1  /*================== destroyQueue ================
2    Deletes all data from a queue and recycles its
3    memory, then deletes & recycles queue head pointer.
4       Pre    Queue is a valid queue
5       Post   All data have been deleted and recycled
6       Return null pointer
7  */
8  QUEUE* destroyQueue (QUEUE* queue)
9  {
10 //Local Definitions
11   QUEUE_NODE* deletePtr;
12
13 //Statements
14   if (queue)
15      {
16       while (queue->front != NULL)
17          {
18           free (queue->front->dataPtr);
19           deletePtr    = queue->front;
20         queue->front = queue->front->next;
21           free (deletePtr);
22          } // while
23       free (queue);
24      } // if
25   return NULL;
26 } // destroyQueue
```

TAD Fila (Queue ADT)
Destruir Fila (Destroy Queue)

- A figura ilustra o primeiro caso para a função Destruir Fila (Destroy Count).



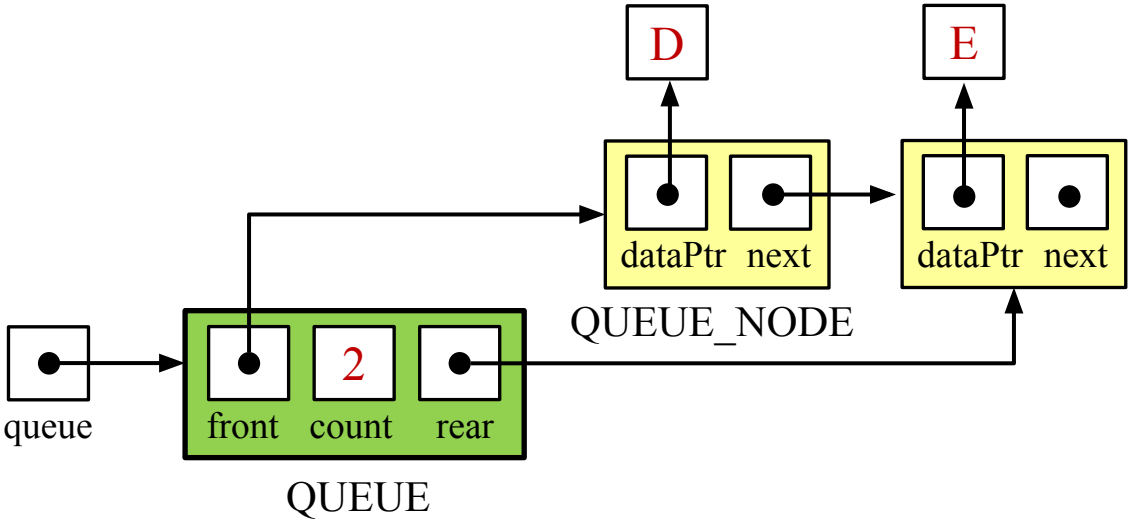

```
free (queue);
return NULL;
```



TAD Fila (Queue ADT)

Destruir Fila (Destroy Queue)

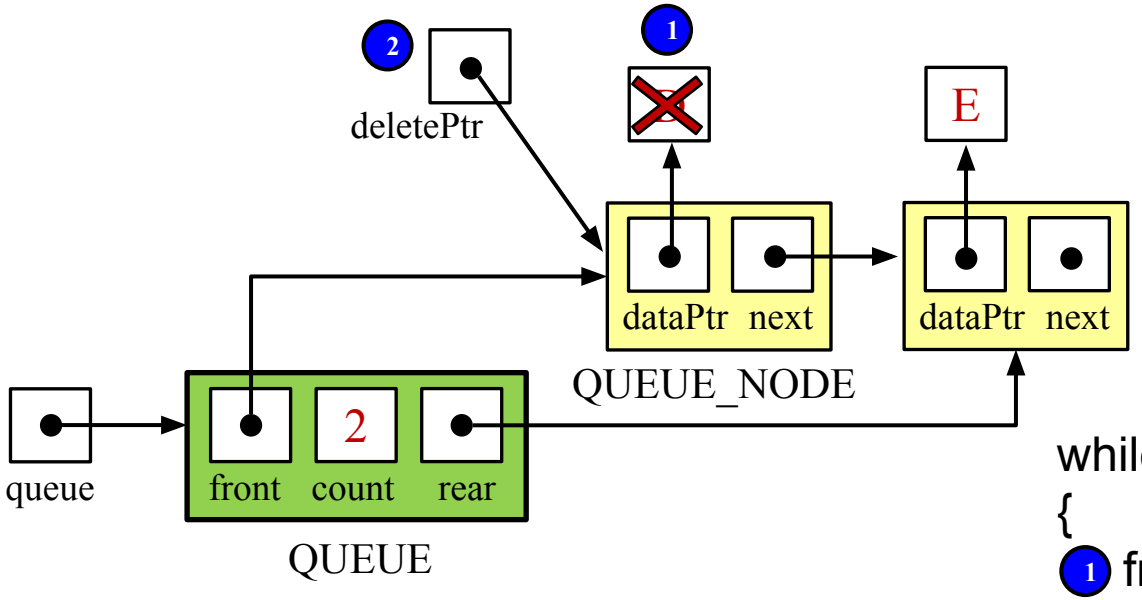
- A figura ilustra o segundo caso para a função Destruir Fila (Destroy Count).



TAD Fila (Queue ADT)

Destruir Fila (Destroy Queue)

- A figura ilustra o segundo caso para a função Destruir Fila (Destroy Count).

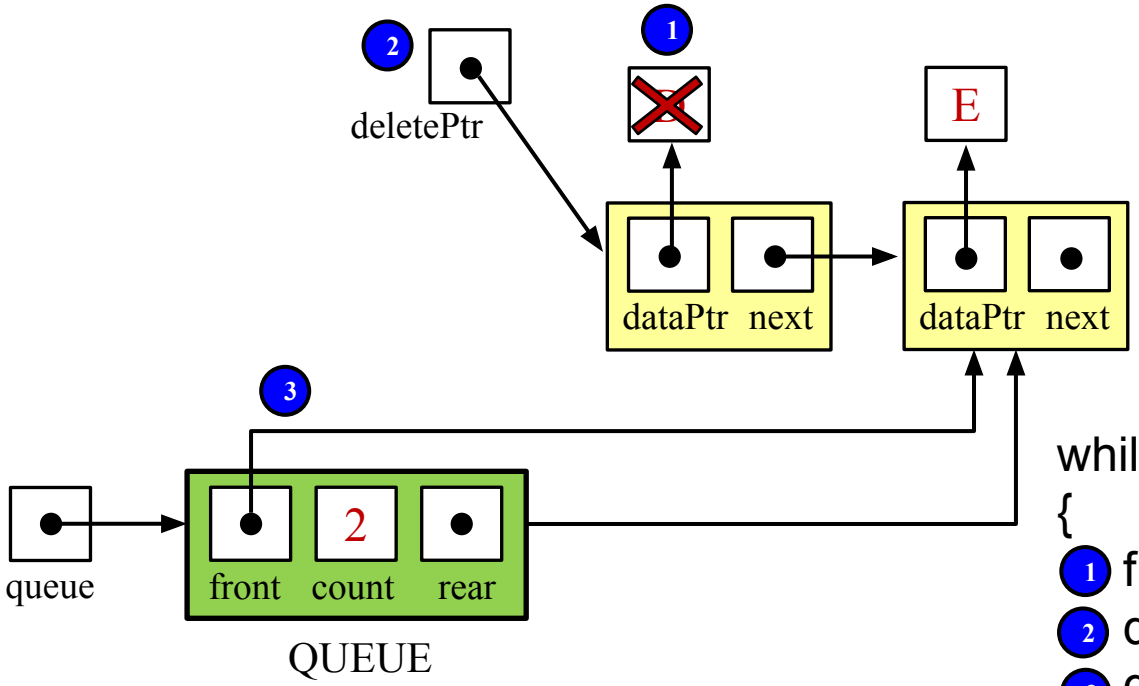



```
while (queue->front != NULL)
{
1 free (queue->front->dataPtr);
2 deletePtr(queue->front);
    queue->front:=queue->front->next;
    free (deletePtr);
}
free (queue);
return NULL;
```

TAD Fila (Queue ADT)

Destruir Fila (Destroy Queue)

- A figura ilustra o segundo caso para a função Destruir Fila (Destroy Count).

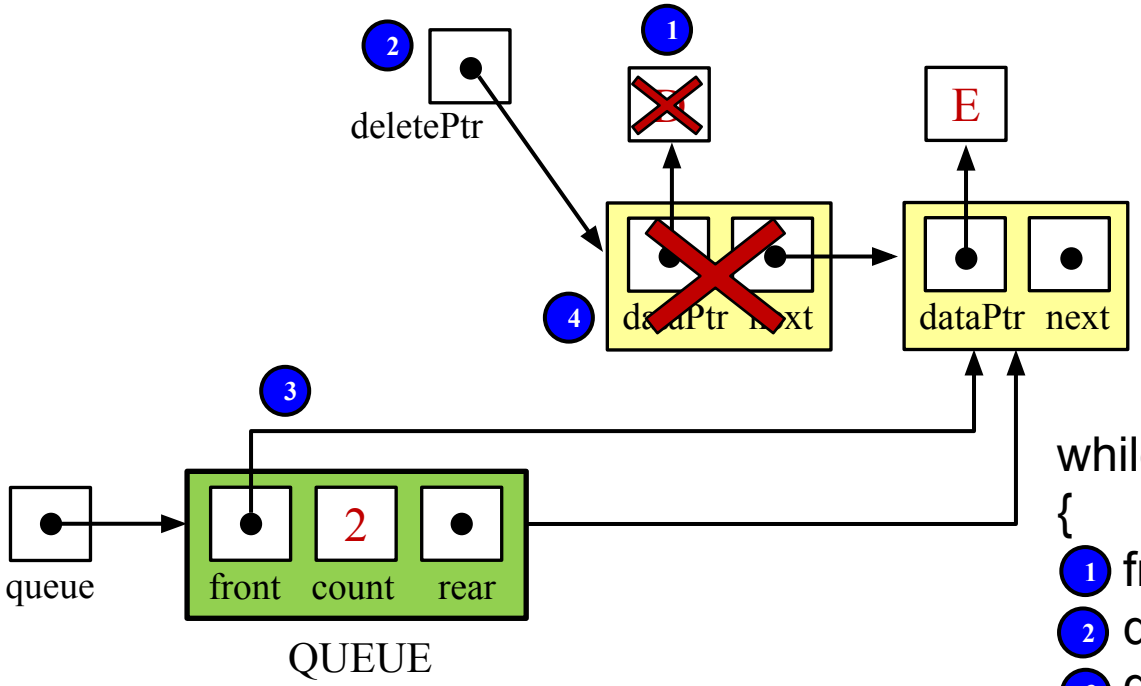



```
while (queue->front != NULL)
{
1 free (queue->front->dataPtr);
2 deletePtr(queue->front);
3 queue->front:=queue->front->next;
  free (deletePtr);
}
free (queue);
return NULL;
```

TAD Fila (Queue ADT)

Destruir Fila (Destroy Queue)

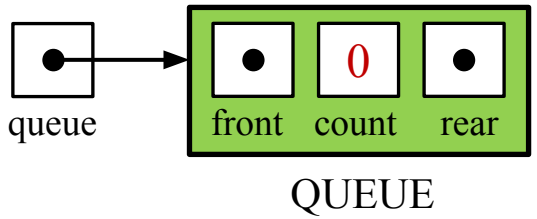
- A figura ilustra o segundo caso para a função Destruir Fila (Destroy Count).



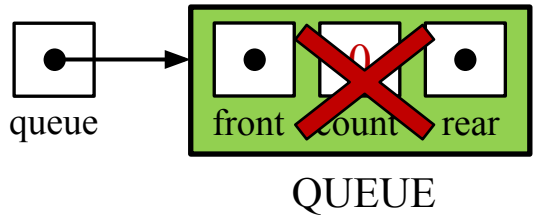

```
while (queue->front != NULL)
{
1 free (queue->front->dataPtr);
2 deletePtr(queue->front);
3 queue->front:=queue->front->next;
4 free (deletePtr);
}
free (queue);
return NULL;
```

TAD Fila (Queue ADT)
Destruir Fila (Destroy Queue)

- A figura ilustra o segundo caso para a função Destruir Fila (Destroy Count).



free (queue);
Return NULL;



Referências

- Gilberg, R.F. e Forouzan, B. A. Data Structures_A Pseudocode Approach with C. Capítulo 4. Queues. Segunda Edição. Editora Cengage, Thomson Learning, 2005.