UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Câmpus Campo Mourão

Ciência da Computação

Algoritmos e Estrutura de Dados 1

Fila com alocação dinâmica

Prof. Rafael Liberato

liberato@utfpr.edu.br

Objetivos

- Entender o funcionamento de uma Fila Dinâmica
- Ser capaz de implementar as operações definidas no TAD Fila manipulando uma estrutura dinâmica de armazenamento.

Roteiro

- TAD Fila
- Fila Dinâmica
- Simulação
- Implementação

TAD Fila

TAD Fila

```
#define ItemType int

typedef struct{

}Queue;

Queue *createQueue ();
void initializeQueue(Queue *q);
int enqueue(Queue * q, ItemType e);
int dequeue(Queue* q, ItemType* e);
int peek(Queue* q, ItemType* e);
int contains(Queue* q, ItemType *e);
int sizeQueue(Queue* q);
int isEmptyQueue(Queue* q);
void printQueue(Queue* q);
```

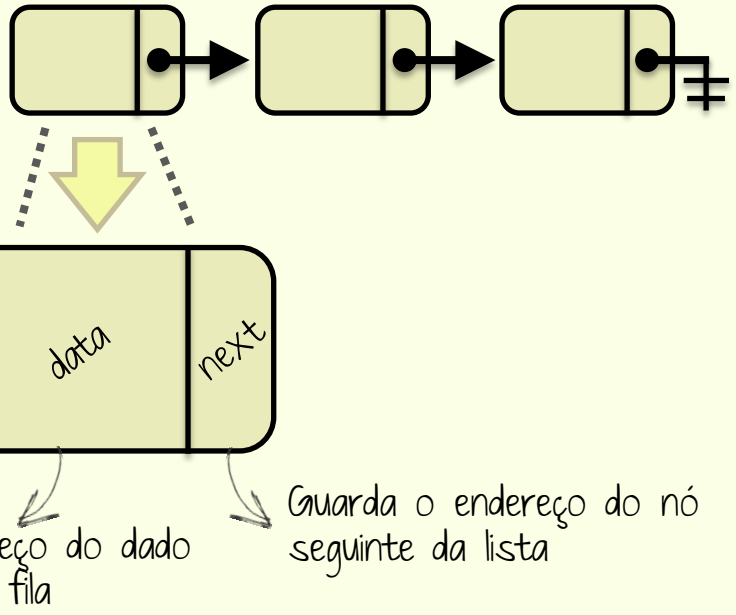
Vamos identificar os atributos que representarão a Fila estática

Estrutura utilizada para armazenar os dados

Fila Dinâmica

Fila Dinâmica

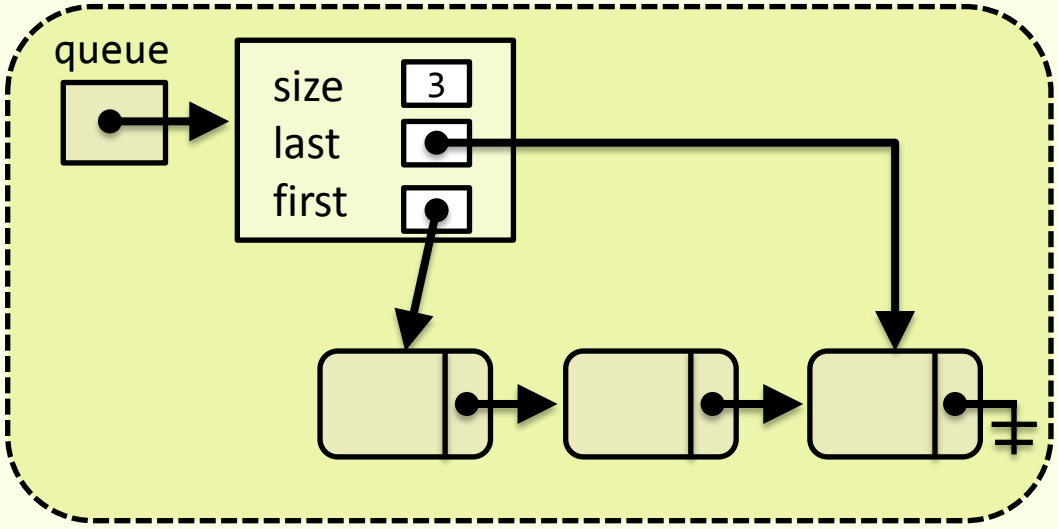
- A **Fila Dinâmica** utiliza uma estrutura de alocação dinâmica de memória para o armazenamento dos dados
- Portanto, temos que utilizar uma estrutura própria para armazenar e interligar os dados
  - ➔ Um encadeamento de nós




```
typedef struct node{
   ItemType    data;
   struct node *next;
}Node;
```

Fila Dinâmica

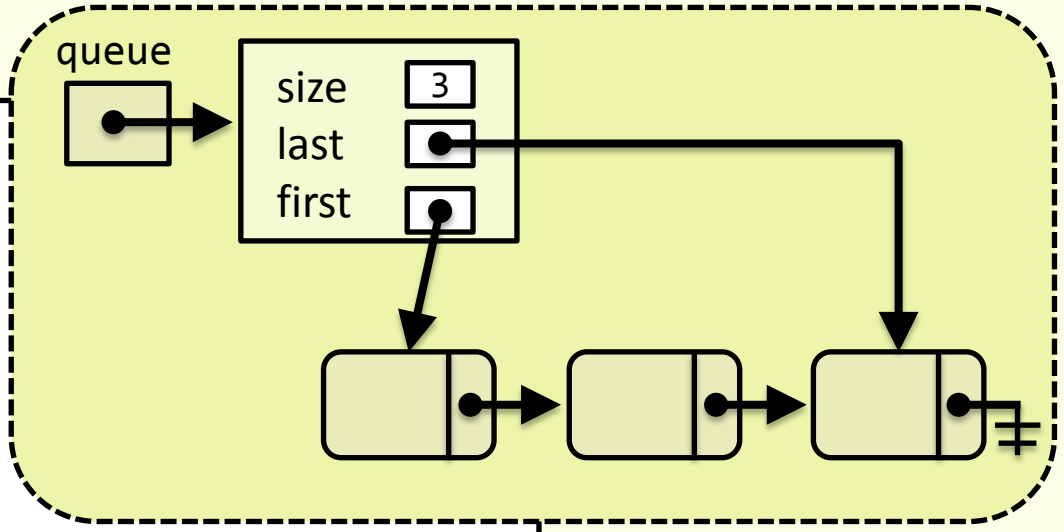
- A Fila Dinâmica é representada pelo endereço do primeiro e do último nó do encadeamento.
- Também utilizaremos um atributo para guardar a quantidade de elementos contidos na Fila.




```
typedef struct node{
    ItemType data;
    struct node *next;
}Node;
```

```
typedef struct{
    Node *first;
    Node *last;
    int size;
}Queue;
```

Fila Dinâmica




```
#define ItemType int

typedef struct{
    Node *first;
    Node *last;
    int size;
}Queue;

Queue *createQueue ();
void initializeQueue(Queue *q);
int enqueue(Queue * q, ItemType e);
int dequeue(Queue* q, ItemType* e);
int peek(Queue* q, ItemType* e);
int contains(Queue* q, ItemType *e);
int sizeQueue(Queue* q);
int isEmptyQueue(Queue* q);
void printQueue(Queue* q);
```

```
typedef struct node{
    ItemType data;
    struct node *next;
}Node;
```

Simulação

Simulação

- Utilize a simulação para entender o comportamento das funções e auxiliá-lo na implementação.

Simulação

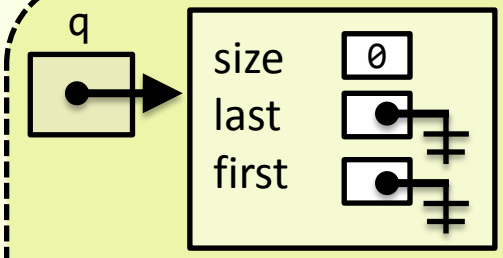
```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```

Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

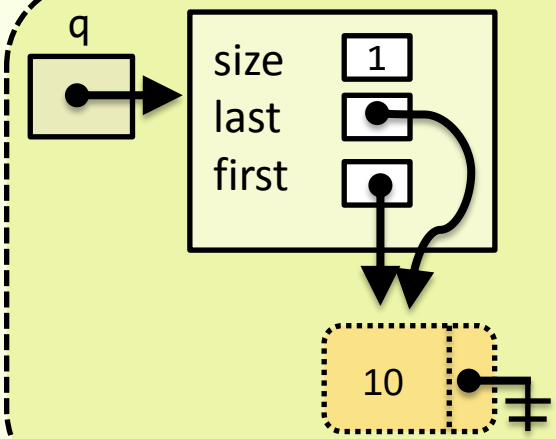
```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

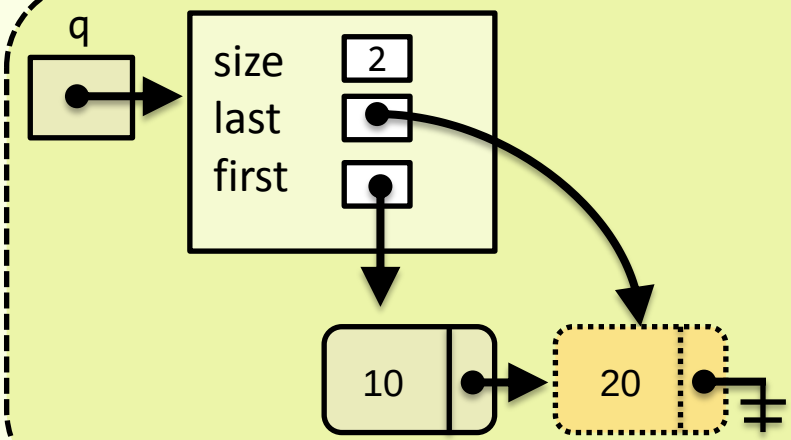
```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

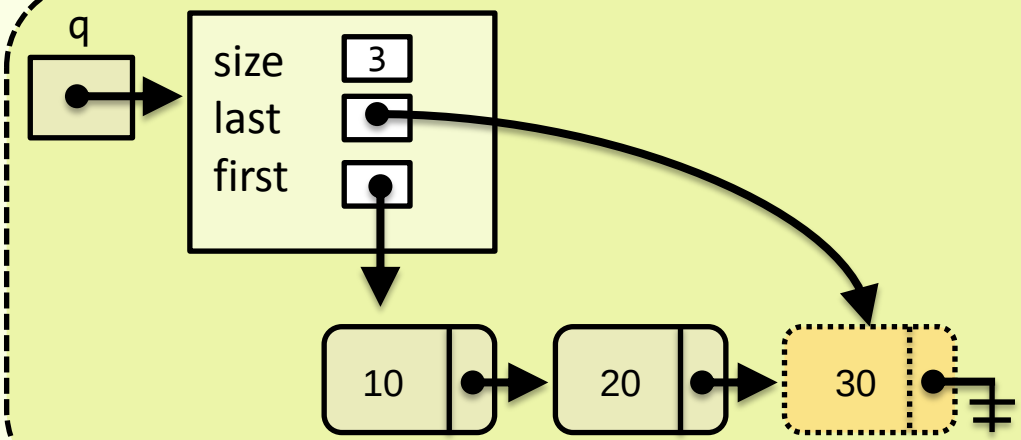
```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

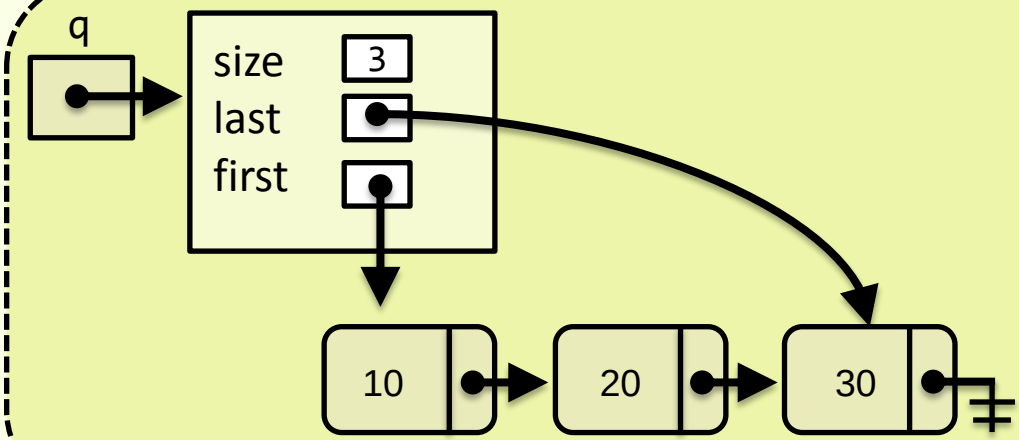
```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

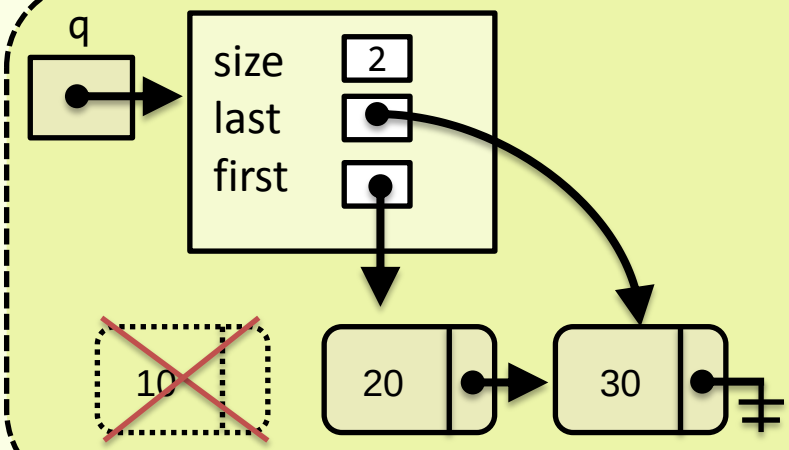
```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



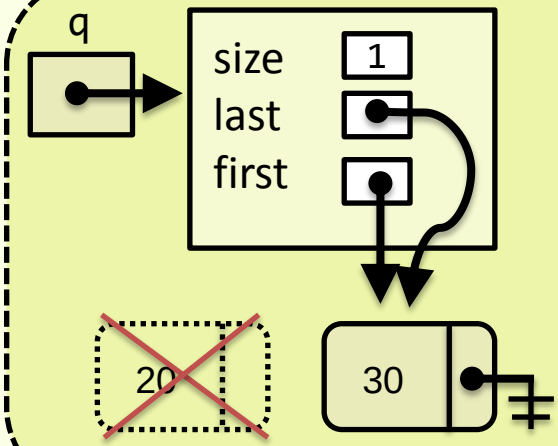
removed

10

Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



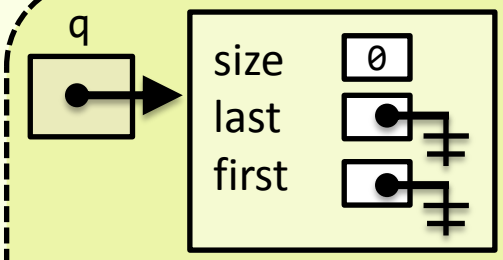
removed

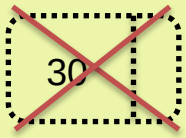
20

Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```





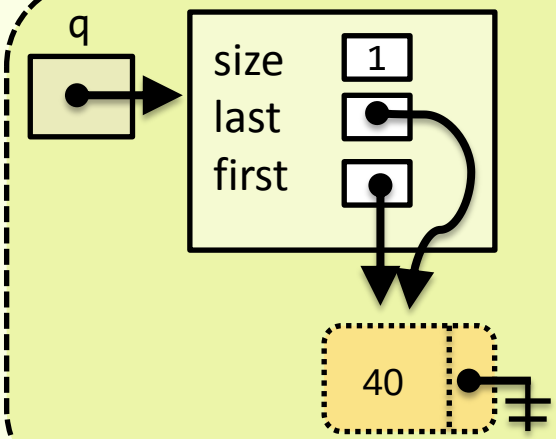
removed

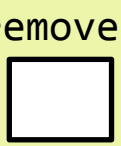
30

Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```

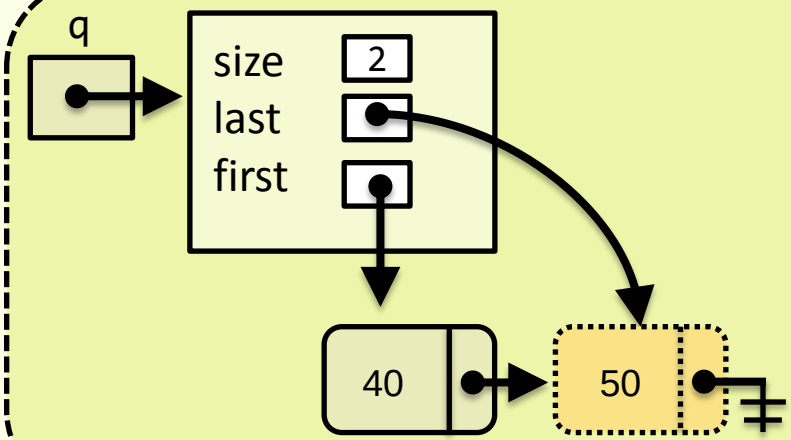




Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

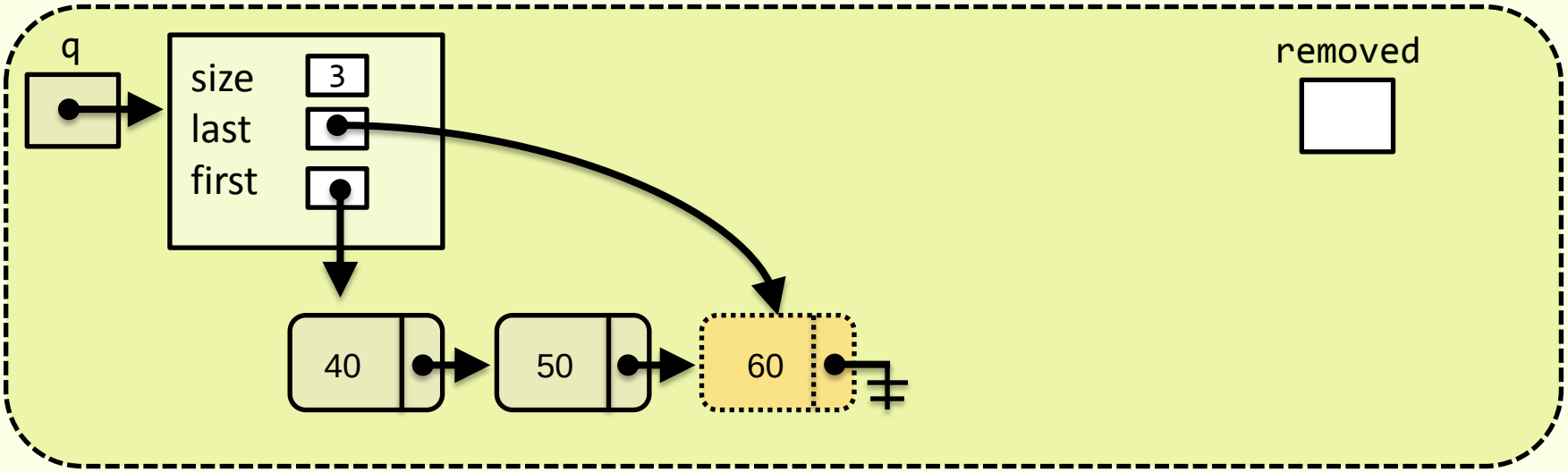
```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

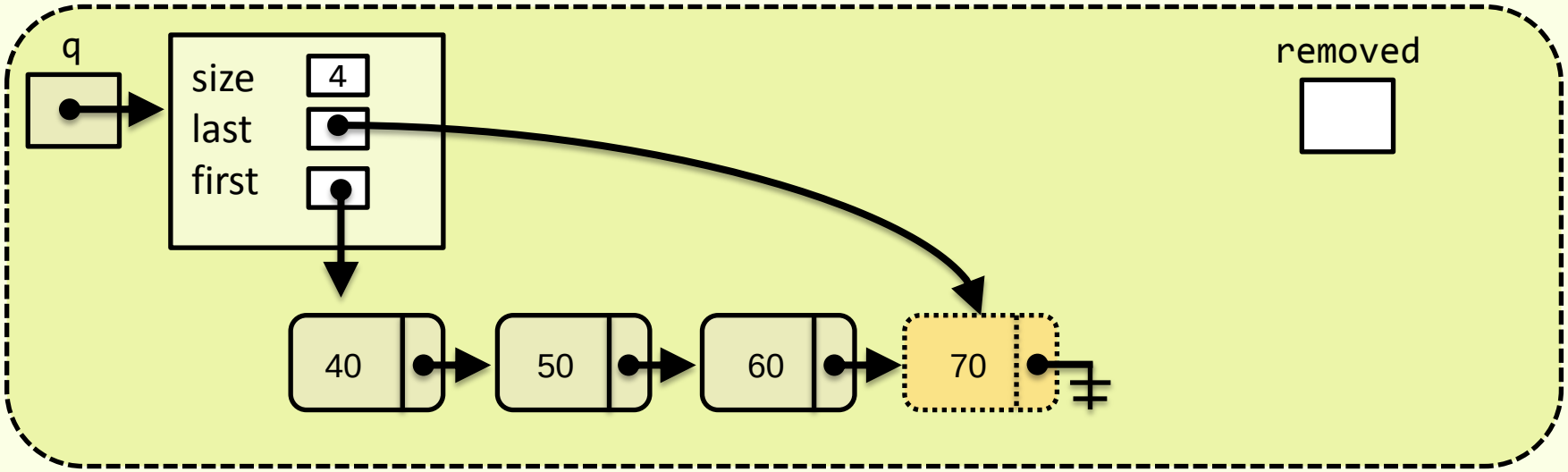
```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Simulação

```
Queue *q = createQueue();
enqueue(q,10);
enqueue(q,20);
enqueue(q,30);
ItemType removed;
dequeue(q, &removed);
dequeue(q, &removed);
dequeue(q, &removed);
```

```
enqueue(q,40);
enqueue(q,50);
enqueue(q,60);
enqueue(q,70);
```



Implementação

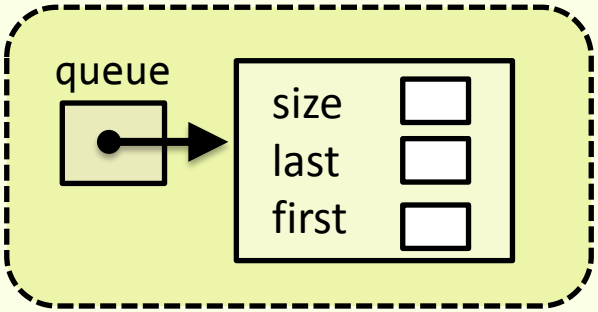
Implementação

- A partir dessa simulação é possível extrair o comportamento das funções sobre os atributos da **Fila dinâmica**

```
Queue *createQueue ();
void initializeQueue(Queue *q);
int enqueue(Queue * q, ItemType e);
int dequeue(Queue* q, ItemType* e);
int peek(Queue* q, ItemType* e);
int contains(Queue* q, ItemType *e);
int sizeQueue(Queue* q);
int isEmptyQueue(Queue* q);
void printQueue(Queue* q);
```

```
typedef struct node{
    ItemType data;
    struct node *next;
}Node;

typedef struct{
    Node *first;
    Node *last;
    int size;
}Queue;
```



Implementação

LET'S DO IT

Referências