ED1 – Fila: Implementação com Alocação Dinâmica

Prof. Andres J. Porfirio

UTFPR

Sumário

- Implementação Dinâmica
  - Código Base
  - dynamic_queue_create
  - dynamic_queue_free
  - dynamic_queue_is_empty
  - dynamic_queue_get_first
  - dynamic_queue_add_last
  - dynamic_queue_remove_first
  - dynamic_queue_get_size

Código Base

- Pode ser modificado o código da pilha dinâmica;
- A Struct principal precisa ser renomeada:

```
typedef char Data;

#define ErrorCode -1

typedef struct item Item;

struct item
{
	Data data;
	Item* next_item;
};
```

definicao do codigo de erro:
NULL para ponteiros
-1 para tipos primitivos

```
typedef struct stack {
    int size;
    Item* first;
}DynamicQueue;
```

"size" será apenas um contador, assim como na fila estática

Note que agora manteremos referência para o primeiro item

".h" e Makefile disponíveis nos anexos da aula

Código Base

- Pode ser modificado o código da pilha dinâmica;
- Para melhor organização, as funções também serão renomeadas:

```
DynamicQueue* dynamic_queue_create();

void dynamic_queue_free(DynamicQueue *q);

int dynamic_queue_is_empty(DynamicQueue *q);

Data dynamic_queue_get_first(DynamicQueue *q);

int dynamic_queue_add_last(DynamicQueue *q, Data d);

Data dynamic_queue_remove_first(DynamicQueue *q);

int dynamic_queue_get_size(DynamicQueue* q);
```

dynamic_queue_create

- Função que inicializa a fila:
  - Alocar memória para a struct principal;
  - Fazer com que o primeiro item aponte para NULL;
  - Atribuir zero ao tamanho;
  - Retornar a estrutura recém alocada;

dynamic_queue_free

- Função que libera a memória após o uso da estrutura:
  - Enquanto existirem elementos na Fila, remova-os;
    - Fique atento à elementos do tipo ponteiro, se a estrutura não trabalha com tipos primitivos então os ponteiros para "Data" também devem ser liberados!
  - Liberar a estrutura principal;

dynamic_queue_is_empty

- Função que verifica se a Fila está vazia:
  - Duas abordagens podem ser utilizadas:
    - **Consultando a variável "size"**:
      - Se o tamanho da fila for zero:
        - Retorne 1;
      - Senão:
        - Retorne 0;
    - **Sem usar a variável "size"**:
      - Se o ponteiro "first" é igual a NULL:
        - Retorne 1;
      - Senão:
        - Retorne 0;

dynamic_queue_get_first

- Função que retorna o primeiro elemento sem removê-lo:
  - Se a fila está vazia:
    - Emitir mensagem de erro;
    - Retornar um **código de erro**: -1 quando "Data" é um tipo primitivo, NULL quando ponteiro;
  - Senão:
    - Retornar o dado do primeiro item;

ErrorCode, Definido no “.h”

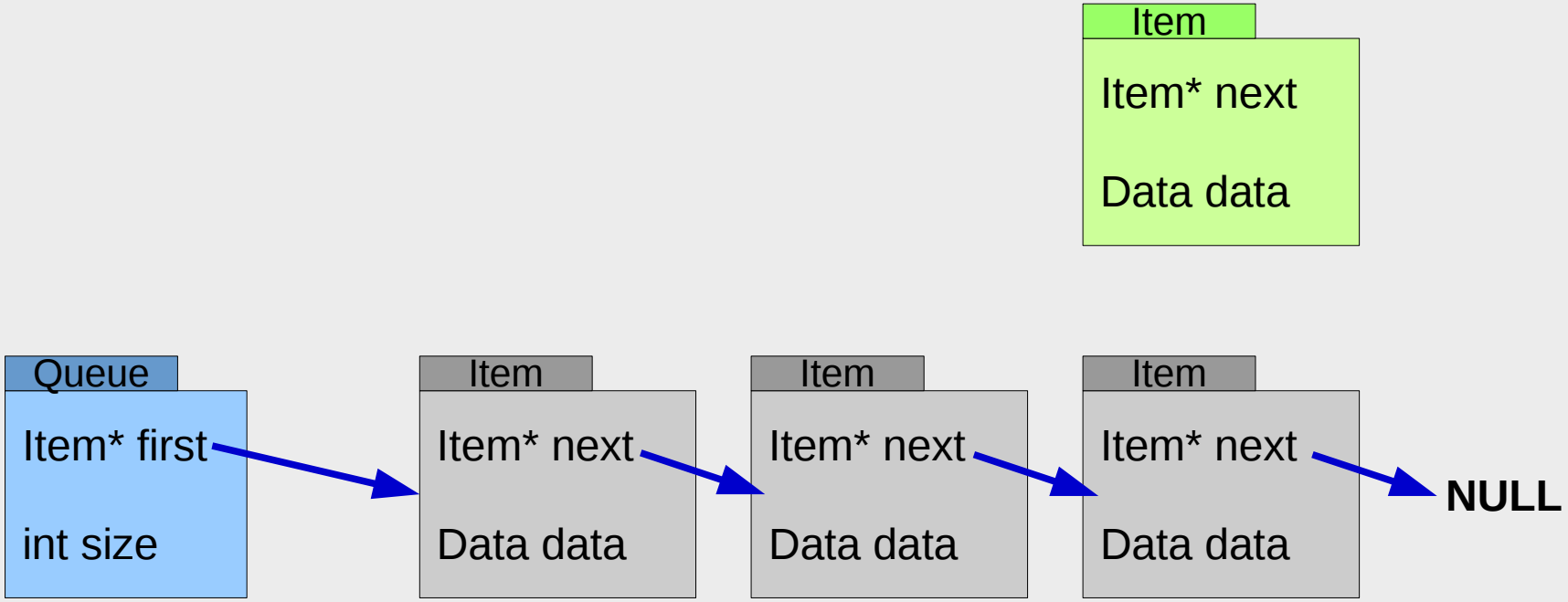
Lembrar: O tipo de retorno da função sempre será “**Data**”, também definido no “.h”

dynamic_queue_add_last

- Função que adiciona um item no final da fila:
  - Alocar um novo item:
    - Verifique se a alocação deu certo, retorne 0 caso "new_item" seja igual a NULL após o "malloc";
  - Configurá-lo de modo que *`new_item->next_item`* receba *`NULL`* e *`new_item->data`* receba o *novo dado*;
  - Se a fila está vazia:
    - Faça com que o ponteiro "first" da estrutura principal aponte para o novo item;
  - Senão:
    - Com um **ponteiro auxiliar**, percorra até o "último" elemento (aquele cujo ponteiro "next_item" é NULL);
    - Faça com que o ponteiro "next_item" do último elemento aponte para o novo item;
  - Incremente o tamanho da fila e retorne 1;

dynamic_queue_add_last

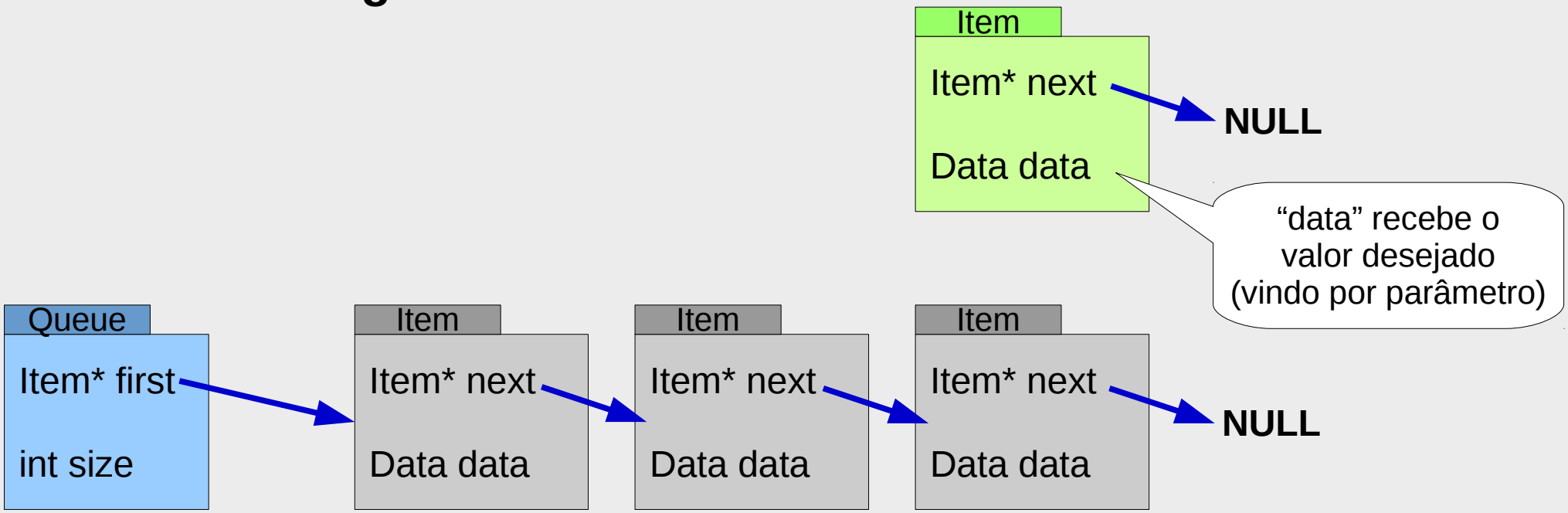
- Função que adiciona um item no final da fila:
    1. **Alocar um novo item**



dynamic_queue_add_last

- Função que adiciona um item no final da fila:

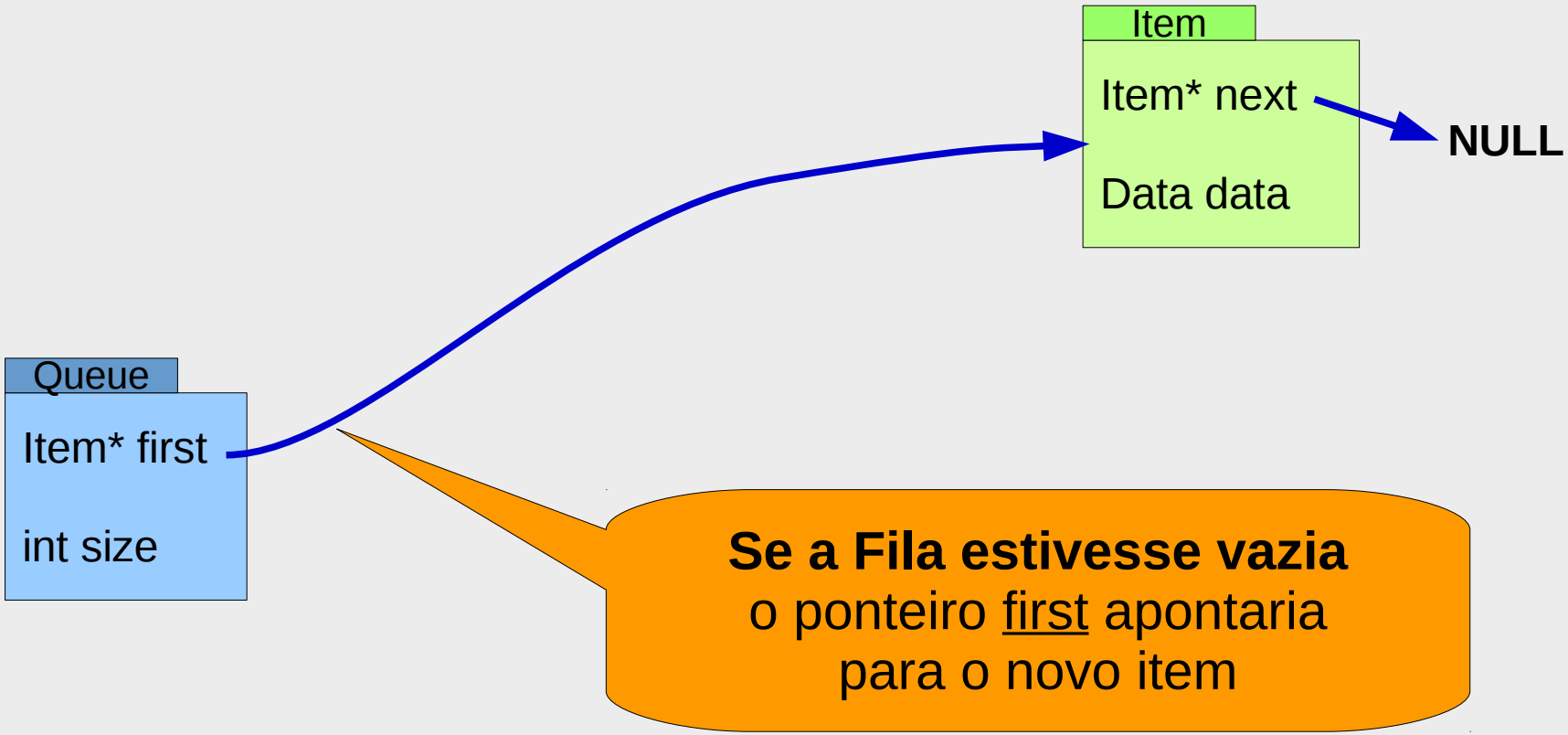
    2. **Configurar o novo item**



dynamic_queue_add_last

- Função que adiciona um item no final da fila:

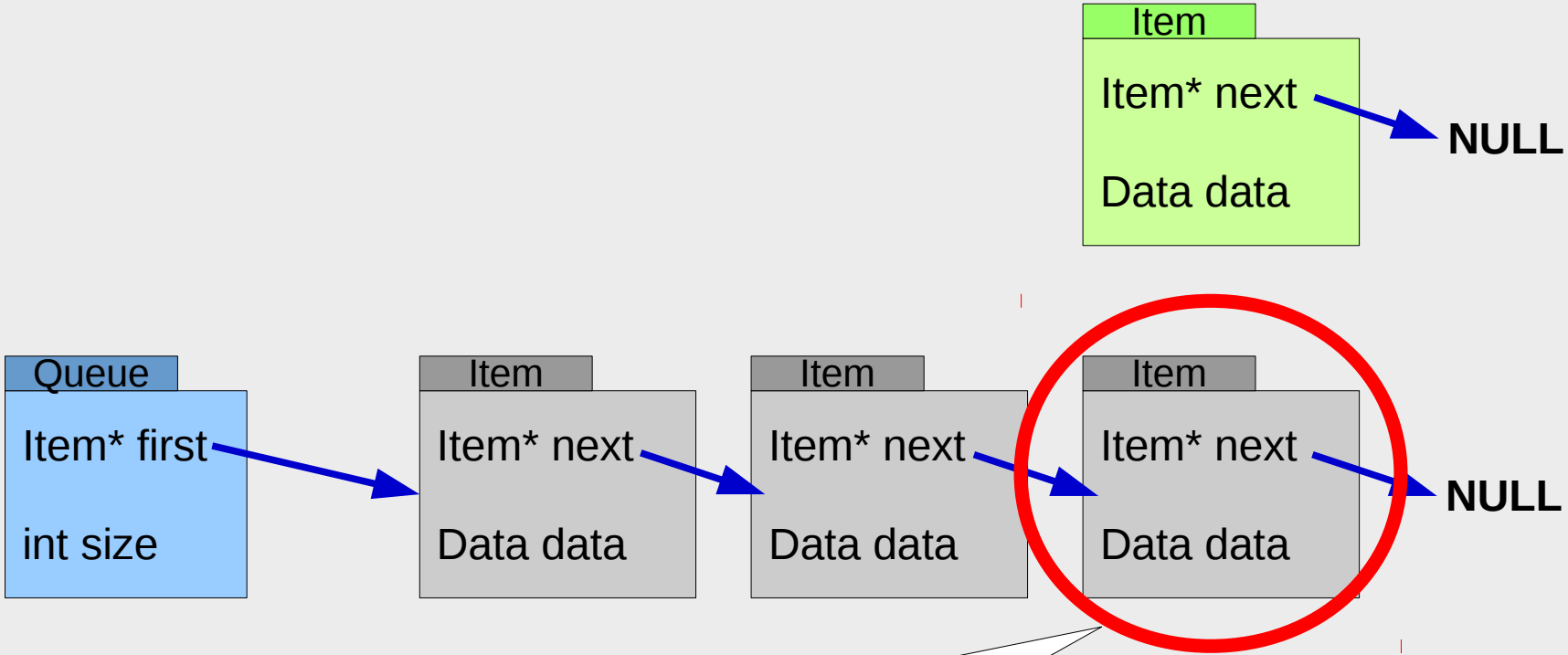
    3. **Posicionar o novo item**



dynamic_queue_add_last

- Função que adiciona um item no final da fila:

    3. **Posicionar o novo item**

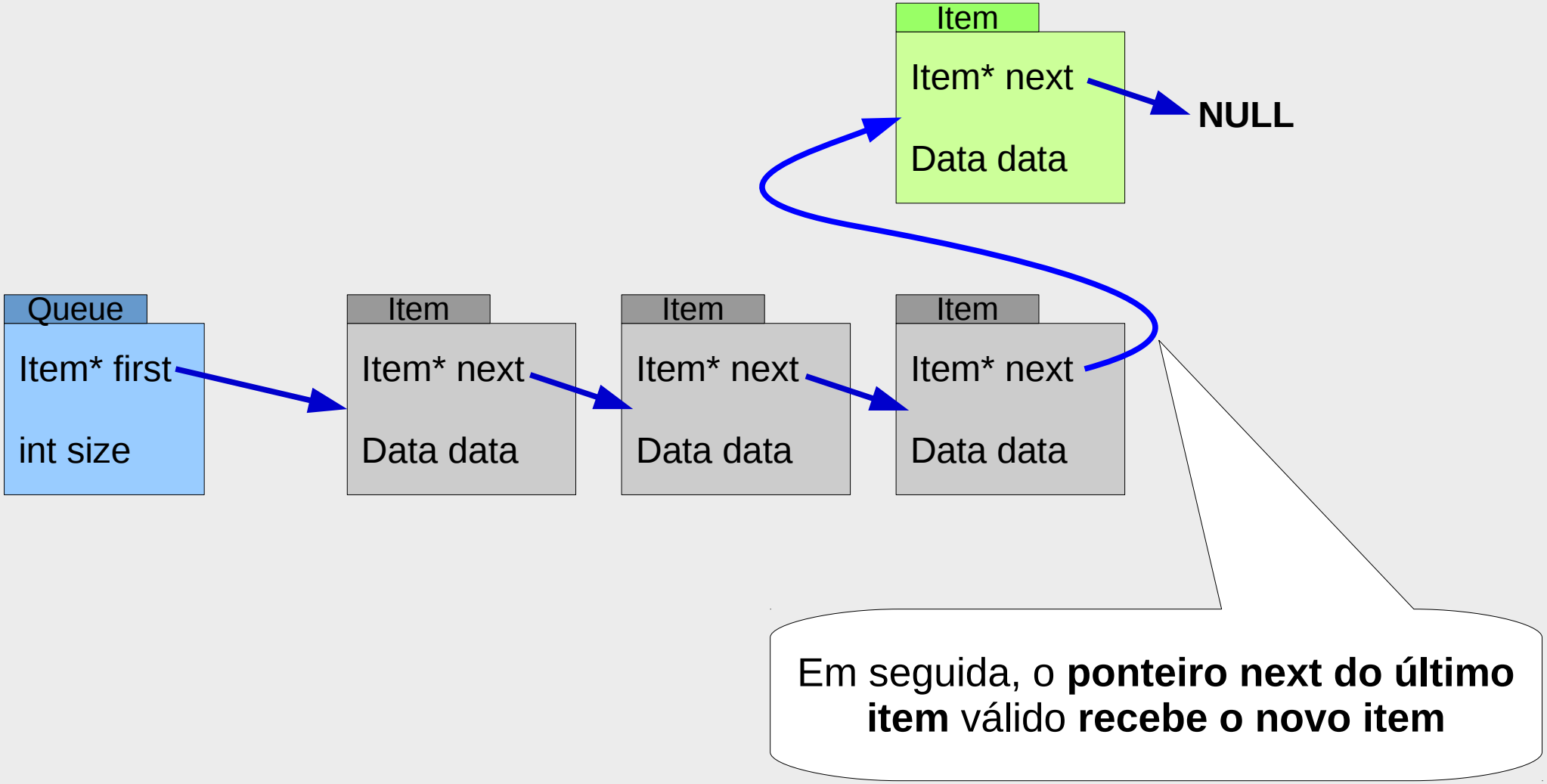


Como a Fila não está vazia
um ponteiro temporário percorre a estrutura em busca do **último item válido**

dynamic_queue_add_last

- Função que adiciona um item no final da fila:

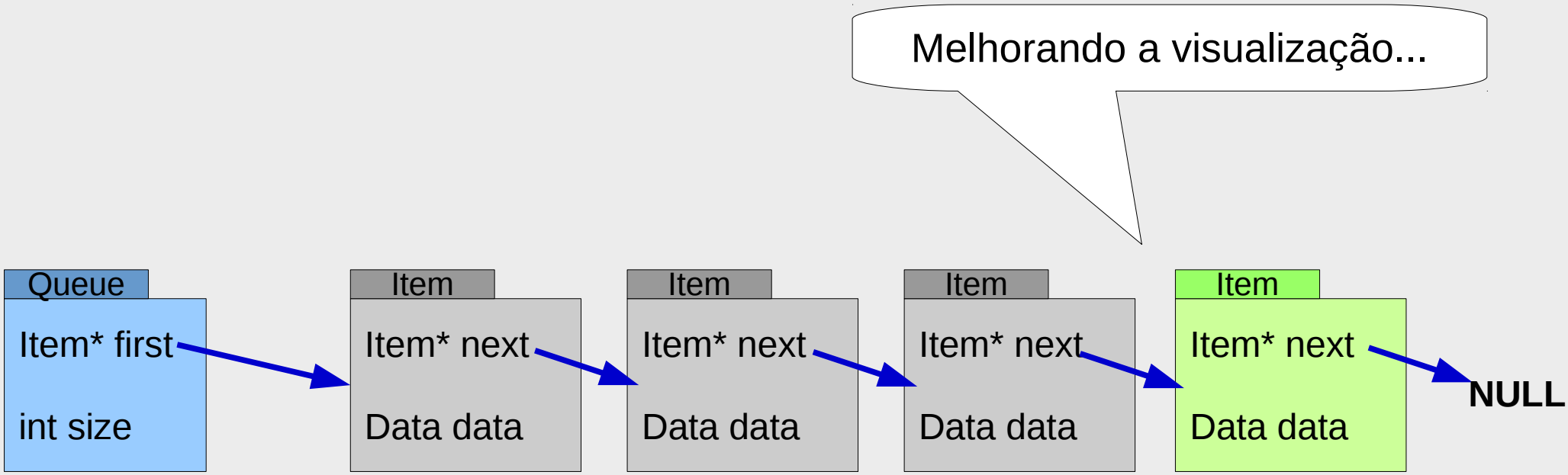
    3. **Posicionar o novo item**



dynamic_queue_add_last

- Função que adiciona um item no final da fila:

    3. **Posicionar o novo item**



Não esquecer de **incrementar o contador**

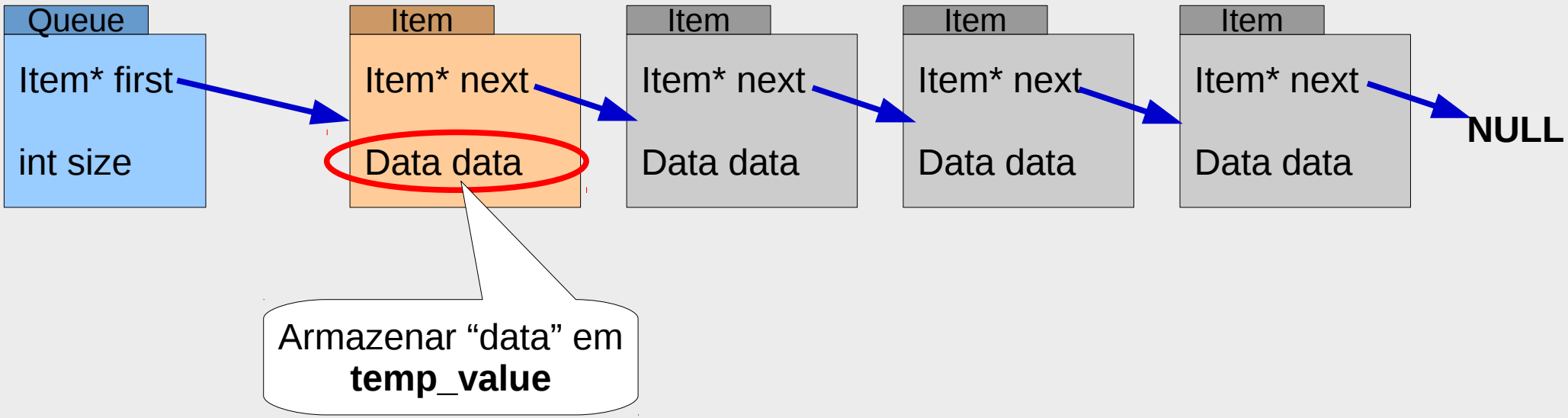
dynamic_queue_remove_first

- Função que remove e retorna o primeiro elemento da fila:
  - Se a fila está vazia:
    - Retorne **ErrorCode** (assim como em .._get_first);
  - Armazene o dado do primeiro item em uma variável "temp_value";
  - Armazene o segundo item (first->next_item) em uma variável "temp";
  - Libere da memória o primeiro elemento;
  - Faça com que o ponteiro "first" da estrutura principal aponte para o segundo item ("temp");
  - Decremente o tamanho, retorne o dado ("temp_value").

dynamic_queue_remove_first

- Função que remove e retorna o primeiro elemento da fila:

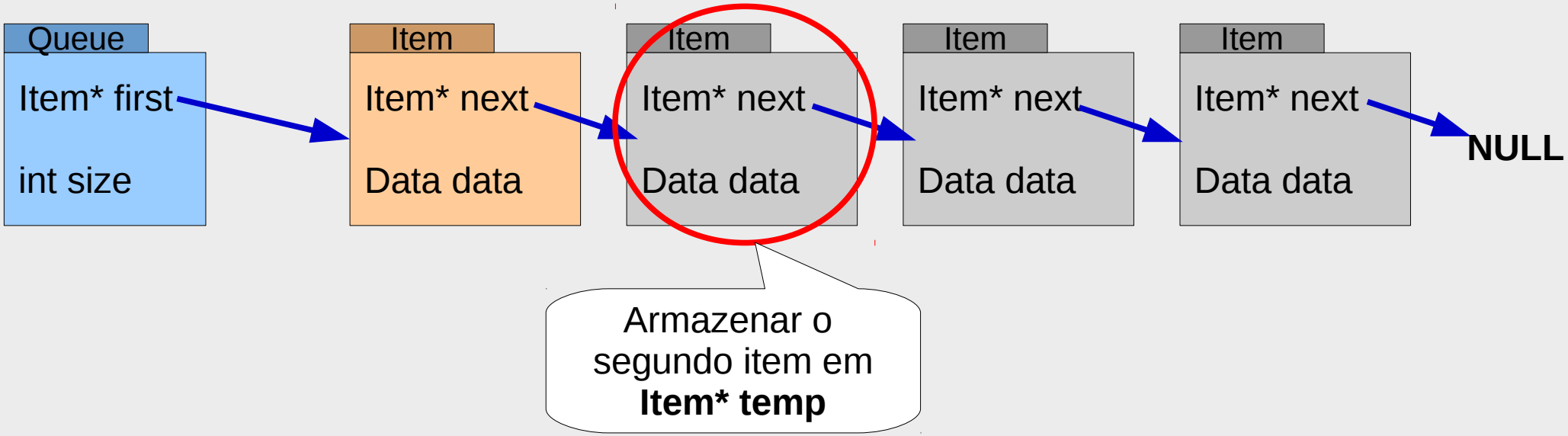
  1. **Caso exista um item a ser removido, armazenar seu valor em uma variável temporária "temp_value"**



dynamic_queue_remove_first

- Função que remove e retorna o primeiro elemento da fila:

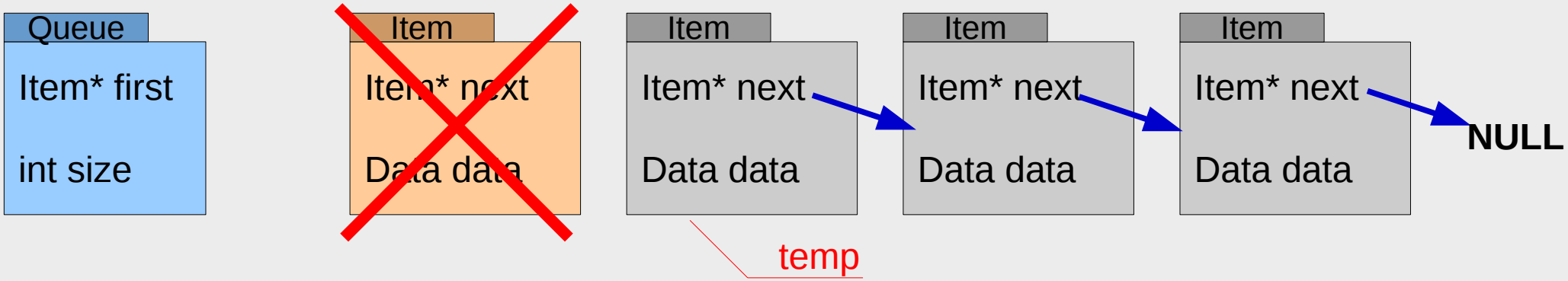
  2. **Armazenar ponteiro do segundo item em uma variável temporária "temp"**



dynamic_queue_remove_first

- Função que remove e retorna o primeiro elemento da fila:

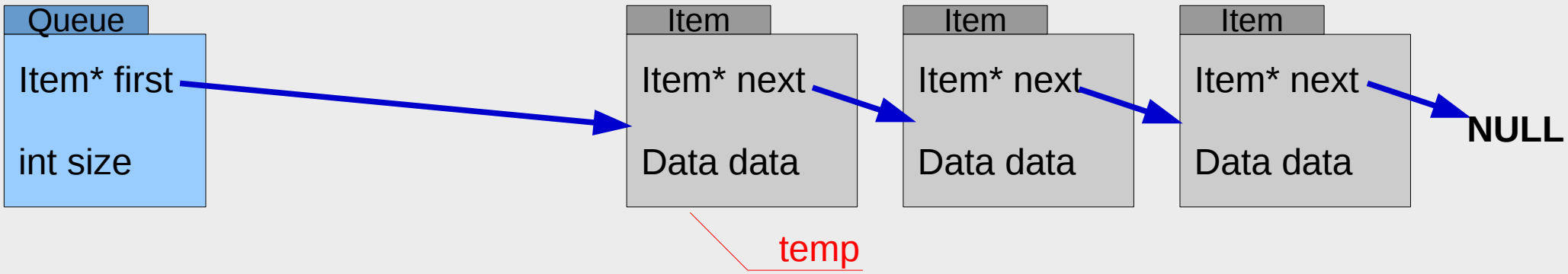
    3. **Liberar a memória do primeiro item**



dynamic_queue_remove_first

- Função que remove e retorna o primeiro elemento da fila:

  3. **Fazer com que o ponteiro first aponte para o segundo item (armazenado em "temp")**



Não esquecer de **decrementar o contador**

dynamic_queue_get_size

- Função que retorna a quantidade de itens na estrutura:
  - Retorne o tamanho;

Exercício

- Com base nos anexos da aula, implemente uma fila com alocação dinâmica;
- Nota: utilize o ".h" fornecido;
- Também está sendo fornecido um código "main" para testes, cuja saída é:

```
adding 'x': sucess
adding 'y': sucess
adding 'z': sucess
adding 'w': sucess
adding 'u': sucess
adding 'o': sucess
removed: x
removed: y
adding 'a': sucess
adding 's': sucess
adding 'd': success
```

//continua...

```
removed: z
adding 'f': sucess
adding 'g': sucess
removed: w
adding 'k': sucess
removed: u
adding 'l': sucess
removed: o
removed: a
removed: s
removed: d
removed: f
removed: g
size: 2
```

Note que todos os *prints* são realizados na função *main*, a Fila não imprime nada. Reproduza essa característica em sua implementação.

Referências

- Alguns conceitos desta aula foram baseados no material do Prof. Eleandro Maschio.