Circular Queue

v0

Generated by Doxygen 1.9.1

Chapter 1

Main Page

Code addapted from https://www.simplilearn.com/tutorials/data-structure-tutorial/circular-qu
uses malloc, realloc and free

To help students get familiar with the use of the structure

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 circular_queue Struct Reference

```
#include <circular_queue_dynamic.h>
```

Public Attributes

- unsigned int max_size
- int front
- int rear
- double ∗ tab

4.1.1 Detailed Description

Addapted from `https://www.simplilearn.com/tutorials/data-structure-tutorial/circular-queue-` uses malloc, realloc and free

Representation of the struct storing a circular queue

4.1.2 Member Data Documentation

4.1.2.1 front

```
int circular_queue::front
```

4.1.2.2 max_size

```
unsigned int circular_queue::max_size
```

current max_size of the queue (starts with 2)

4.1.2.3 rear

```
int circular_queue::rear
```

index of the front and rear (both -1 if empty)

4.1.2.4 tab

```
double* circular_queue::tab
```

pointer to a block with max_size elements

The documentation for this struct was generated from the following file:

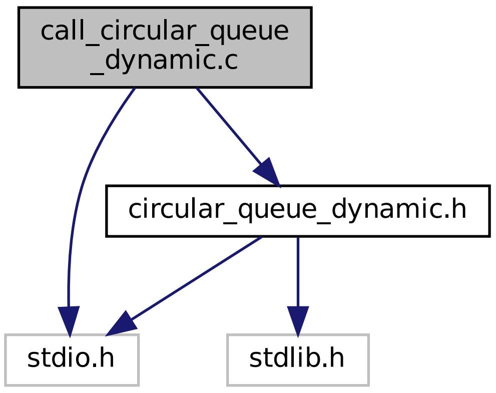
- circular_queue_dynamic.h

Chapter 5

File Documentation

5.1 call_circular_queue_dynamic.c File Reference

```
#include <stdio.h>
#include "circular_queue_dynamic.h"
```
Include dependency graph for call_circular_queue_dynamic.c:



Functions

- int main ()

5.1.1 Function Documentation

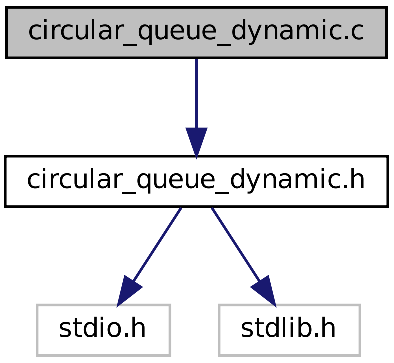
5.1.1.1 main()

```
int main ( )
```

5.2 circular_queue_dynamic.c File Reference

```
#include "circular_queue_dynamic.h"
```

Include dependency graph for circular_queue_dynamic.c:



Functions

- int inic (struct circular_queue ∗q)
- void freeQ (struct circular_queue ∗q)
- int isEmpty (const struct circular_queue ∗q)
- int isFull (const struct circular_queue ∗q)
- int expand (struct circular_queue ∗q)
- int enQ (struct circular_queue ∗q, double val)
- int deQ (struct circular_queue ∗q, double ∗val)
- void printQ (const struct circular_queue ∗q)

5.2.1 Function Documentation

5.2.1.1 deQ()

```
int deQ (
            struct circular_queue * q,
            double * val )
```

Dequeues the front elemento of the queue

Parameters

q	pointer to the queue
val	value that was at the front of the queue

Returns

1 if the queue was not empty, otherwise it returns 0 indicating this function should not have been called

5.2.1.2 enQ()

```
int enQ (
            struct circular_queue * q,
            double val )
```

Enqueues a new elemento to the queue

Parameters

q	pointer to the queue
val	value to me added to the end of the queue

Returns

1 if value was successuly added, otherwise returns 0 indicating queue was full and the call to expand failed to enlarge the queue.

5.2.1.3 expand()

```
int expand (
            struct circular_queue * q )
```

Doubles the allocated space, but only if the queue is full

Parameters

q	pointer to the queue to enlarged

Returns

-1 if the queue is not full (does nothing), if memory allocation failed returns 0 (false) otherwise returns 1 (true)

5.2.1.4 freeQ()

```
void freeQ (
            struct circular_queue * q )
```

Frees space allocated by the queue

Parameters

q	pointer to the queue to be cleared - the allocated memory is released

5.2.1.5 inic()

```
int inic (
            struct circular_queue * q )
```

Inciializes queue q - this is the first routine that must be called before using q.

Parameters

q	pointer to the queue to be inicialized

Returns

if memory allocation failed returns 0 (false) otherwise returns 1 (true)

5.2.1.6 isEmpty()

```
int isEmpty (
            const struct circular_queue * q )
```

Informs is the queue is empty

Parameters

q	pointer to the queue to be evaluated

Returns

1 (true) if ∗q is empty otherwise returns 0 (false)

5.2.1.7 isFull()

```
int isFull (
             const struct circular_queue * q )
```

Informs is the queue is full

Parameters

q	pointer to the queue to be evaluated

Returns

1 (true) if ∗q is full otherwise returns 0 (false)

5.2.1.8 printQ()

```
void printQ (
             const struct circular_queue * q )
```

Prints the value of all elements in the queue (for DEBUG purposes)

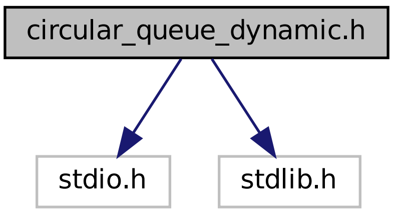
Parameters

q	pointer to the queue

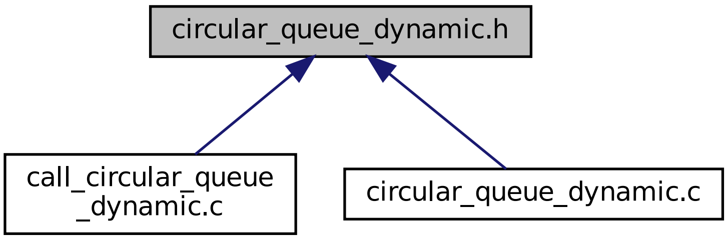
5.3 circular_queue_dynamic.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for circular_queue_dynamic.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct circular_queue

Functions

- int inic (struct circular_queue ∗q)
- void freeQ (struct circular_queue ∗q)
- int isEmpty (const struct circular_queue ∗q)
- int isFull (const struct circular_queue ∗q)
- int expand (struct circular_queue ∗q)
- int enQ (struct circular_queue ∗q, double val)
- int deQ (struct circular_queue ∗q, double ∗val)
- void printQ (const struct circular_queue ∗q)

5.3.1 Function Documentation

5.3.1.1 deQ()

```
int deQ (
            struct circular_queue ∗ q,
            double ∗ val )
```

Dequeues the front elemento of the queue

Parameters

q	pointer to the queue
val	value that was at the front of the queue

Returns

1 if the queue was not empty, otherwise it returns 0 indicating this function should not have been called

5.3.1.2 enQ()

```
int enQ (
            struct circular_queue * q,
            double val )
```

Enqueues a new elemento to the queue

Parameters

q	pointer to the queue
val	value to me added to the end of the queue

Returns

1 if value was successuly added, otherwise returns 0 indicating queue was full and the call to expand failed to enlarge the queue.

5.3.1.3 expand()

```
int expand (
            struct circular_queue * q )
```

Doubles the allocated space, but only if the queue is full

Parameters

q	pointer to the queue to enlarged

Returns

-1 if the queue is not full (does nothing), if memory allocation failed returns 0 (false) otherwise returns 1 (true)

5.3.1.4 freeQ()

```
void freeQ (
            struct circular_queue * q )
```

Frees space allocated by the queue

Parameters

q	pointer to the queue to be cleared - the allocated memory is released

5.3.1.5 inic()

```
int inic (
            struct circular_queue * q )
```

Inciializes queue q - this is the first routine that must be called before using q.

Parameters

q	pointer to the queue to be inicialized

Returns

if memory allocation failed returns 0 (false) otherwise returns 1 (true)

5.3.1.6 isEmpty()

```
int isEmpty (
            const struct circular_queue * q )
```

Informs is the queue is empty

Parameters

q	pointer to the queue to be evaluated

Returns

1 (true) if ∗q is empty otherwise returns 0 (false)

5.3.1.7 isFull()

```
int isFull (
            const struct circular_queue * q )
```

Informs is the queue is full

Parameters

q	pointer to the queue to be evaluated

Returns

1 (true) if ∗q is full otherwise returns 0 (false)

5.3.1.8 printQ()

```
void printQ (
             const struct circular_queue * q )
```

Prints the value of all elements in the queue (for DEBUG purposes)

Parameters

q	pointer to the queue

Index