

Circular Queue

v0

Generated by Doxygen 1.9.1

| | |
|--------------------------------------------------|-----------|
| 1 Main Page | 1 |
| 2 Class Index | 3 |
| 2.1 Class List | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Class Documentation | 7 |
| 4.1 circular_queue Struct Reference | 7 |
| 4.1.1 Detailed Description | 7 |
| 4.1.2 Member Data Documentation | 7 |
| 4.1.2.1 front | 7 |
| 4.1.2.2 max_size | 7 |
| 4.1.2.3 rear | 8 |
| 4.1.2.4 tab | 8 |
| 5 File Documentation | 9 |
| 5.1 call_circular_queue_dynamic.c File Reference | 9 |
| 5.1.1 Function Documentation | 9 |
| 5.1.1.1 main() | 10 |
| 5.2 circular_queue_dynamic.c File Reference | 10 |
| 5.2.1 Function Documentation | 10 |
| 5.2.1.1 deQ() | 10 |
| 5.2.1.2 enQ() | 11 |
| 5.2.1.3 expand() | 11 |
| 5.2.1.4 freeQ() | 12 |
| 5.2.1.5 inic() | 12 |
| 5.2.1.6 isEmpty() | 12 |
| 5.2.1.7 isFull() | 13 |
| 5.2.1.8 printQ() | 13 |
| 5.3 circular_queue_dynamic.h File Reference | 13 |
| 5.3.1 Function Documentation | 14 |
| 5.3.1.1 deQ() | 14 |
| 5.3.1.2 enQ() | 15 |
| 5.3.1.3 expand() | 15 |
| 5.3.1.4 freeQ() | 15 |
| 5.3.1.5 inic() | 16 |
| 5.3.1.6 isEmpty() | 16 |
| 5.3.1.7 isFull() | 16 |
| 5.3.1.8 printQ() | 17 |
| Index | 19 |

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:

| | |
|------------------------------------------|---|
| circular_queue | 7 |
|------------------------------------------|---|

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

| | |
|-----------------------------------------------|----|
| call_circular_queue_dynamic.c | 9 |
| circular_queue_dynamic.c | 10 |
| circular_queue_dynamic.h | 13 |

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:

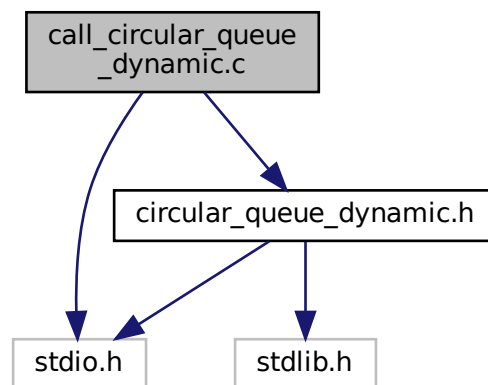
- [circular_queue_dynamic.h](#)

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

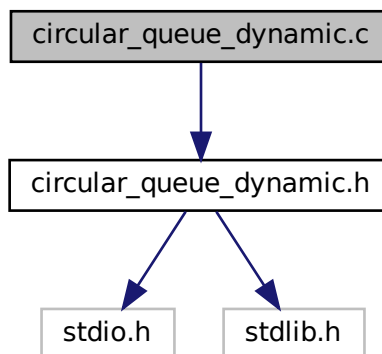
5.1.1.1 main()

```
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 element of the queue

Parameters

| | |
|------------|------------------------------------------|
| <i>q</i> | pointer to the queue |
| <i>val</i> | 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

| | |
|------------|-------------------------------------------|
| <i>q</i> | pointer to the queue |
| <i>val</i> | 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

| | |
|----------|----------------------------------|
| <i>q</i> | 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

| | |
|----------|-----------------------------------------------------------------------|
| <i>q</i> | pointer to the queue to be cleared - the allocated memory is released |
|----------|-----------------------------------------------------------------------|

5.2.1.5 inic()

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

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

Parameters

| | |
|----------|----------------------------------------|
| <i>q</i> | 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

| | |
|----------|--------------------------------------|
| <i>q</i> | 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

| | |
|----------|--------------------------------------|
| <i>q</i> | 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)

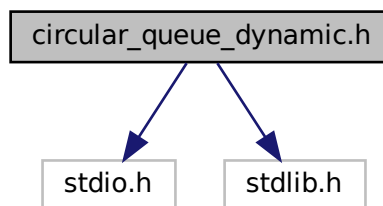
Parameters

| | |
|----------|----------------------|
| <i>q</i> | pointer to the queue |
|----------|----------------------|

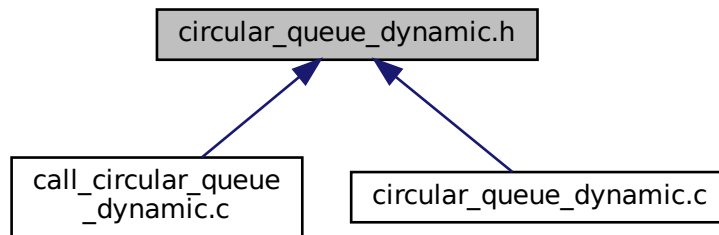
5.3 circular_queue_dynamic.h File Reference

```
#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 element of the queue

Parameters

| | |
|------------|------------------------------------------|
| <i>q</i> | pointer to the queue |
| <i>val</i> | 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

| | |
|------------|-------------------------------------------|
| <i>q</i> | pointer to the queue |
| <i>val</i> | 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

| | |
|----------|----------------------------------|
| <i>q</i> | 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

| | |
|----------|-----------------------------------------------------------------------|
| <i>q</i> | pointer to the queue to be cleared - the allocated memory is released |
|----------|-----------------------------------------------------------------------|

5.3.1.5 inic()

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

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

Parameters

| | |
|----------|----------------------------------------|
| <i>q</i> | pointer to the queue to be initialized |
|----------|----------------------------------------|

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 if the queue is empty

Parameters

| | |
|----------|--------------------------------------|
| <i>q</i> | 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 if the queue is full

Parameters

| | |
|----------|--------------------------------------|
| <i>q</i> | 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

| | |
|----------|----------------------|
| <i>q</i> | pointer to the queue |
|----------|----------------------|

Index

- call_circular_queue_dynamic.c, [9](#)
 - main, [9](#)
- circular_queue, [7](#)
 - front, [7](#)
 - max_size, [7](#)
 - rear, [7](#)
 - tab, [8](#)
- circular_queue_dynamic.c, [10](#)
 - deQ, [10](#)
 - enQ, [11](#)
 - expand, [11](#)
 - freeQ, [11](#)
 - inic, [12](#)
 - isEmpty, [12](#)
 - isFull, [12](#)
 - printQ, [13](#)
- circular_queue_dynamic.h, [13](#)
 - deQ, [14](#)
 - enQ, [15](#)
 - expand, [15](#)
 - freeQ, [15](#)
 - inic, [16](#)
 - isEmpty, [16](#)
 - isFull, [16](#)
 - printQ, [17](#)
- deQ
 - circular_queue_dynamic.c, [10](#)
 - circular_queue_dynamic.h, [14](#)
- enQ
 - circular_queue_dynamic.c, [11](#)
 - circular_queue_dynamic.h, [15](#)
- expand
 - circular_queue_dynamic.c, [11](#)
 - circular_queue_dynamic.h, [15](#)
- freeQ
 - circular_queue_dynamic.c, [11](#)
 - circular_queue_dynamic.h, [15](#)
- front
 - circular_queue, [7](#)
- inic
 - circular_queue_dynamic.c, [12](#)
 - circular_queue_dynamic.h, [16](#)
- isEmpty
 - circular_queue_dynamic.c, [12](#)
 - circular_queue_dynamic.h, [16](#)
- isFull
 - circular_queue_dynamic.c, [12](#)
 - circular_queue_dynamic.h, [16](#)
- main
 - call_circular_queue_dynamic.c, [9](#)
- max_size
 - circular_queue, [7](#)
- printQ
 - circular_queue_dynamic.c, [13](#)
 - circular_queue_dynamic.h, [17](#)
- rear
 - circular_queue, [7](#)
- tab
 - circular_queue, [8](#)