

[返回](#)

8-1 Queue (35分)

Design a generic circular queue class. using C++ standard exception class: overflow_error and underflow_error in the design.

the picture below shows the UML class diagram for the generic queue class.

CQueue<T>		
-queueSize:	const int	/*the queue size, is a const member, can store up to queueSize-1 datas*/
-head:	int	/*Record the subscript of the queue head*/
-rear:	int	/*Record the subscript of the end of the queue*/
-data_buff:	T*	/*Data storage buffer*/
+CQueue(): /*the default queue size is 10*/		
+CQueue(int s)		
+~CQueue()		
+getSize():	int const	/*mean: the function return int, is a const member function*/
+getNumbers():	int const	/* Calculation formula for the total number of queue elements is : (rear - head + queueSize)%queueSize */
+getHead():	T	
+enqueue(value. i):	void	
+deQueue():	T	
+isEmpty():	bool const	
+isFull():	bool const	
+show():	void const	

The test code is in test.cpp, The output is (There is a space at the end of the line): now the queue is full! 0 1 2 3 4 5 6 7 8 0 5 6 7 8

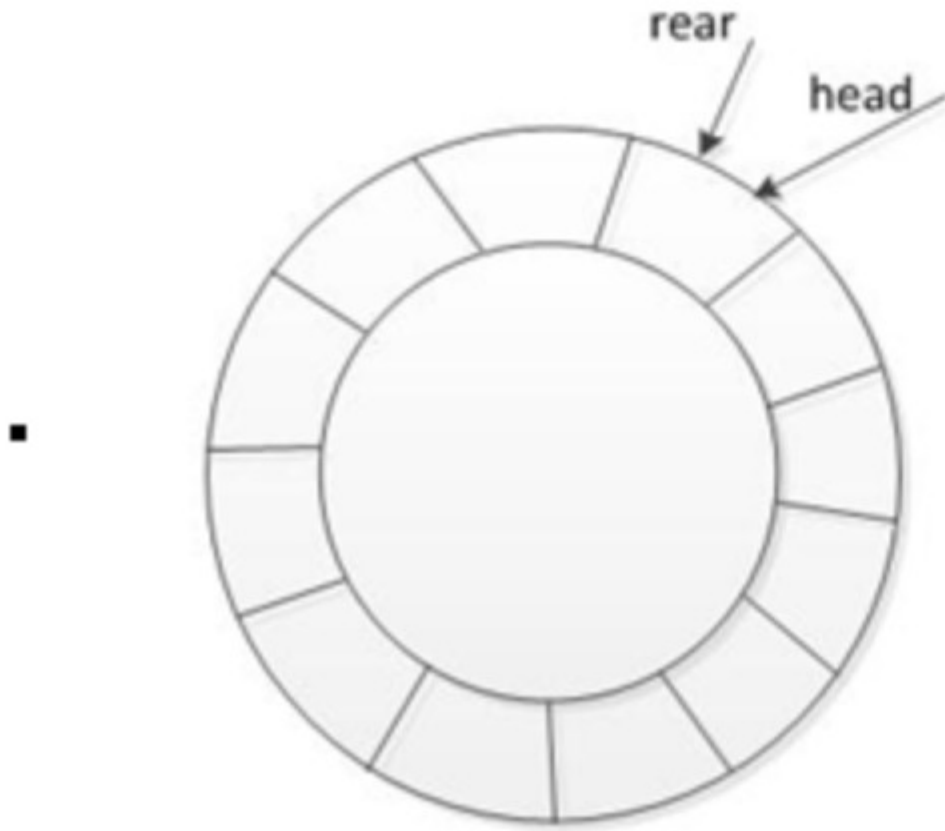


Figure 1 rear==head, The queue is empty

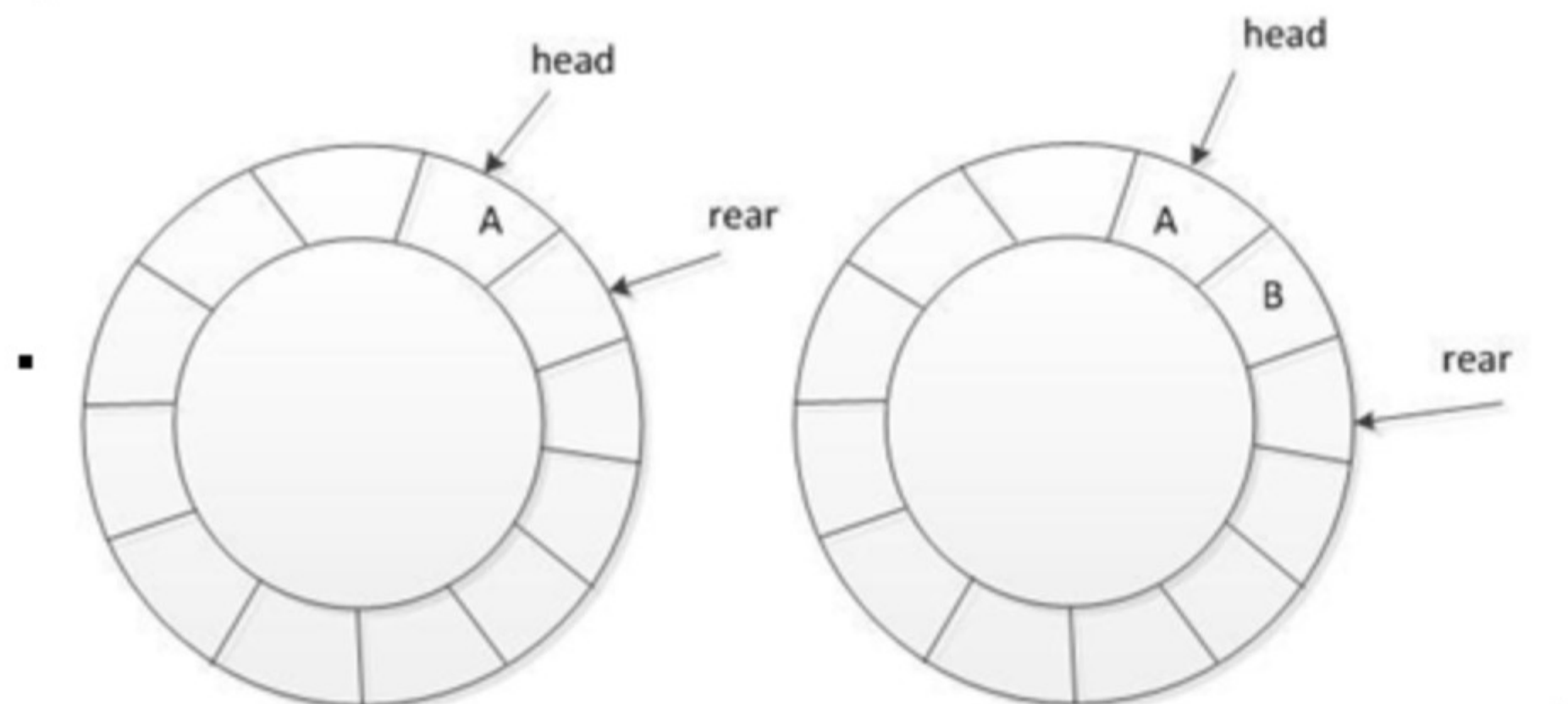


Figure 2 Add elements 'A' and 'B' to the queue in turn

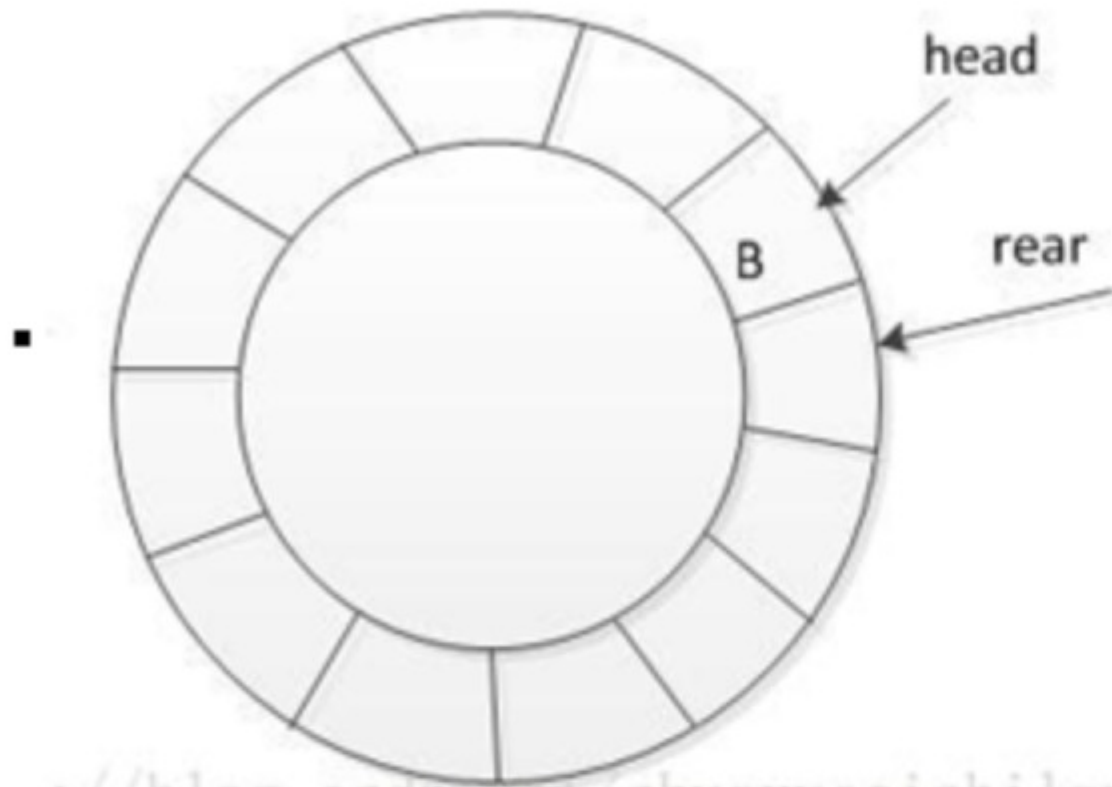


Figure 3 Put the element 'A' of the head out of the queue

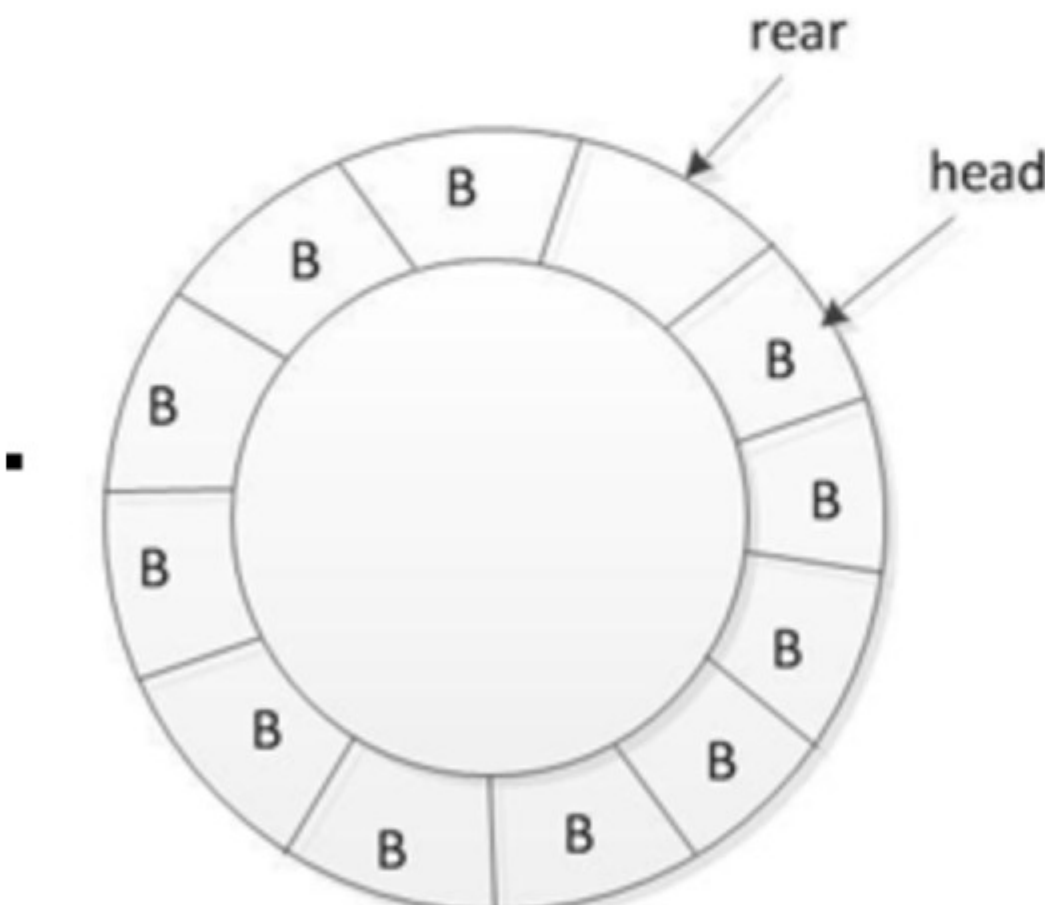


Figure 4 The queue is full when (rear+1)%QueueSize == front

```
/*test.cpp*/
#include <iostream>
#include <stdexcept>
using namespace std;
#include "CQueue.h"
int main()
{
    try {
        CQueue<double> rq;
        for (int i = 0; i < rq.getSize()-1; i++)
            rq.enqueue(i);
        if (rq.isFull()) printf("now the queue is full! ");
        if (!rq.isEmpty()) rq.show();
        cout << rq.getHead() << " ";
        for (int i = 0; i < 5; i++) // dequeuing 5 elements
            rq.deQueue();
        rq.show();
    }
    catch (overflow_error& r)
    {
        cout << r.what();
    }
    catch (underflow_error& r)
    {
        cout << r.what();
    }
    return 0;
}
```

作者 胡兰青
单位 浙江大学