1. 循环队列

```
#pragma once
#define MAXSIZE 5
#define FALSE 0
#define TRUE 1
template<class T>
class SeqQueue
{
public:
	SeqQueue();	//构造函数
	~SeqQueue();	//析构函数
	void EnterSeqQueue(T x); //入队操作，插入元素 x 作为队尾元素
	void PrintSeqQueue();	//打印队中元素
	void DeleteSeqQueue();	//出队操作，删除队头元素
	T GetSeqHead();	//返回队头元素
	int SeqLength();	//队列长度
	bool SeqIsFull();	//判断是否为满
	bool SeqIsEmpty();	//判断是否为空
public:
	T * data;	// 初始化的分配动态储存空间
	int front;	//头指针
	int rear;	//尾指针
};


#include<iostream>
#include<string>
using namespace std;


template<class T>
SeqQueue<T>::SeqQueue()
{
	this->data = new T[MAXSIZE];	//申请空间
	this->rear =this->front= 0;	//指针都置为 0
}


template<class T>
SeqQueue<T>::~SeqQueue()
{
        delete[] data;  //释放空间
}

template<class T>
void SeqQueue<T>::EnterSeqQueue(T x)
{
        if (SeqIsFull())
        {
                cout << "队列已满,插入失败" << endl;
                return;
        }
        else if (!data)
        {
                cout << "队列不存在" << endl;
                return;
        }
        else
        {
                data[rear] = x;    //插入元素
                rear = (rear + 1) % MAXSIZE;  //尾指针移动
        }
}

template<class T>
void SeqQueue<T>::PrintSeqQueue()
{
        if (!data)
        {
                cout << "队列不存在" << endl;
                return;
        }
        else if(rear>front)
        {
                for (int i = front; i < rear; i++)
                {
                        cout << data[i] << "  ";
                }
                cout << endl;
        }

        else
        {
                for (int i = front; i <MAXSIZE; i++)
                {
```

```
			cout << data[i] << "  ";
		}
		for (int i = 0; i <front-1; i++)
		{
			cout << data[i] << "  ";
		}
		cout << endl;
	}
}

template<class T>
void SeqQueue<T>::DeleteSeqQueue()
{
	if (!data)
	{
		cout << "队列不存在" << endl;
		return;
	}
	if (this->rear == this->front)
	{
		cout << "队列为空" << endl;
		return;
	}
	else
	{
		front = (front + 1) % MAXSIZE; //头指针移动
	}
}

template<class T>
T SeqQueue<T>::GetSeqHead()
{
	if (!data)
	{
		cout << "队列不存在" << endl;
		return FALSE;
	}
	else if (this->front == this->rear)
	{
		cout << "队列为空" << endl;
		return FALSE;
	}
	else
	{
		return data[this->front];  //返回队头元素，队头指针不变
	}
}

template<class T>
int SeqQueue<T>::SeqLength()
{
	if (!data)
	{
		cout << "队列不存在" << endl;
		return FALSE;
	}
	return (this->rear - this->front+MAXSIZE)%MAXSIZE;// 队列长度
}

template<class T>
bool SeqQueue<T>::SeqIsFull()
{
	if (this->front == (this->rear+1)%MAXSIZE)
	{
		return 1;//队列为满返回 1
	}

	else
	{
		return 0;//队列不为满返回 0
	}
}
template<class T>
bool SeqQueue<T>::SeqIsEmpty()
{
	if (this->front == this->rear)
	{
		return 1;  //队列为空返回 1
	}
	else
	{
		return 0;//队列不为空返回 0
	}
}

int main()
{
	SeqQueue<int> q;
```

```
    q.EnterSeqQueue(1);
    q.EnterSeqQueue(2);
    q.EnterSeqQueue(3);
    q.EnterSeqQueue(4);

    q.PrintSeqQueue();
    cout<<"队列的长度为： " << q.SeqLength()<<endl;
    cout << "队头元素为： " << q.GetSeqHead() << endl;

    q.DeleteSeqQueue();
    q.DeleteSeqQueue();

    q.PrintSeqQueue();
    cout << "队列的长度为： " << q.SeqLength() << endl;

    cout << "队头元素为： " << q.GetSeqHead() << endl;
    q.EnterSeqQueue(5);
    q.EnterSeqQueue(6);

    q.PrintSeqQueue();
    cout << "队列的长度为： " << q.SeqLength() << endl;


}
```

2.向前复制

```
#ifndef OOP_CPP_MY_QUEUE_H
#define OOP_CPP_MY_QUEUE_H

#include <iostream>
#include <vector>
using namespace std;

template <class T>
class my_queue{
private:
  int length;      //长度
  vector<T> content;  //内容
  int max_len;     //最大长度

public:
  my_queue() = default;
```

```
    explicit my_queue(int _max_len=10);
    T& begin();        //首元素
    T& end();          //尾元素
    int push(T item);   //添加在尾部
    T pop();           //弹出首部
    bool full();       //检验是否已满
    bool empty();      //检验是否为空
    int len();         //输出长度
};


#endif //OOP_CPP_MY_QUEUE_H


#include "my_queue.h"
#include <iostream>
#include <vector>

using namespace std;

template <class T>
my_queue<T>::my_queue(const int _max_len) {
    length = 0;
    this->max_len = _max_len;
    this->content = vector<T>(max_len);
    cout<<"init queue"<<endl;
}

template <class T>
bool my_queue<T>::empty() {
    return length == 0;
}

template <class T>
bool my_queue<T>::full() {
    if(length == max_len){
        return true;
    }
    return false;
}

template <class T>
int my_queue<T>::push(T item) {
    if(!this->full()){
        content[length] = item;
//      cout<<content[length]<<endl;
        length += 1;
        return 0;
    }
    cout<<"queue is full! no more push!"<<endl;
    return -1;
}

template <class T>
T my_queue<T>::pop() {
    if(!this->empty()){
        T item = this->content[0];
        for(int i = 0 ; i < this->length; i++){
            this->content[i] = this->content[i+1];
        }
//      this->content[length] = NULL;
        length -= 1;
        return item;
    }
    cout<<"queue is empty! cannot pop!"<<endl;
//  return NULL;
}

template <class T>
T& my_queue<T>::begin() {
    if(!this->empty()){
        return this->content[0];
    }
    cout<<"queue is empty!"<<endl;
//  return NULL;
}

template <class T>
T& my_queue<T>::end() {
    return this->content[length-1]; // 最后一个元素在 length-1 的位置
}

template <class T>
int my_queue<T>::len() {
    return length;
}
```

```
int main(){
  my_queue<int> a(5);                    // 初始化长度为 5 的队列
  cout<<"is queue empty? "<<(a.empty()?"yes":"no")<<endl;        // 检查是否为空
  a.pop();                               // 空队列取值
  for(int i = 0 ; i < 5 ; i++){
    a.push(i);                           // 装满
    cout<<"add "<<i<<" queue length is "<<a.len()<<" now"<<endl; //返回长度
  }
  cout<<"is queue full? "<<(a.full()?"yes":"no")<<endl;          // 检验是否已满
  a.push(6);                             // 继续装报错
  cout<<"pop first element: "<<a.pop()<<endl;                           // 弹出首元素
  cout<<"begin element is "<<a.begin()<<endl;      // 首元素
  cout<<"end element is "<<a.end()<<endl;          // 末尾元素
}
```