**2022212153 陈祥烨 计科22-2班 第三章作业**

第一题

Queue.hpp

#pragma once

#include<iostream>

using namespace std;

const int maxlen = 100;

template<class T>

class Queue //循环数组队列

{

public:

Queue();

Queue(Queue<T>& Q);

int length()const;

bool empty()const;

bool full()const; //原理：为区别队空，留一个空(rear + 1) % maxlen == front

bool get\_front(T& x)const;

bool append(const T x);

bool serve(); //出队

private:

int count;

int front, rear; //队头，队尾,头指针不含元素

T data[maxlen];

};

template<class T>

Queue<T>::Queue()

{

count = 0;

front = rear = 0;

cout << "Queue默认构造函数调用" << endl;

}

template<class T>

Queue<T>::Queue(Queue<T>& Q)

{

count = Q.count;

front = Q.count;

rear = Q.rear;

for (int i = front + 1; i % maxlen == rear; i++)

{

data[i] = Q.data[i];

}

cout << "Queue拷贝构造函数调用" << endl;

}

template<class T>

int Queue<T>::length()const

{

return count;

}

template<class T>

bool Queue<T>::empty()const

{

//if (count == 0)

// return true;

//return false;

return front == rear;

}

template<class T>

bool Queue<T>::full()const

{

//if (count == maxlen - 1)

// return true;

//return false;

return (rear + 1) % maxlen == front;

}

template<class T>

bool Queue<T>::get\_front(T& x)const

{

if (empty())

return false;

x = data[(front + 1) % maxlen];

return true;

}

template<class T>

bool Queue<T>::append(const T x)

{

if (full())

return false;

rear = (rear + 1) % maxlen;

data[rear] = x;

count++;

return true;

}

template<class T>

bool Queue<T>::serve() //出队

{

if (empty())

return false;

front = (front + 1) % maxlen;

count--;

return false;

}

test.cpp

#include"Queue.hpp"

void test()

{

Queue<int> q;

if (q.empty())

cout << "q is empty" << endl;

for (int i = 0; i < maxlen; i++)

q.append(i);

int front;

q.get\_front(front);

cout << "front of q is : " << front << endl;

if (q.full())

cout << "q is full" << endl;

q.serve();

q.get\_front(front);

cout << "after serve, front of q is : " << front << endl;

if (!q.full())

cout << "q is not full" << endl;

Queue<int> p(q);

q.get\_front(front);

cout << "front of p is : " << front << endl;

}

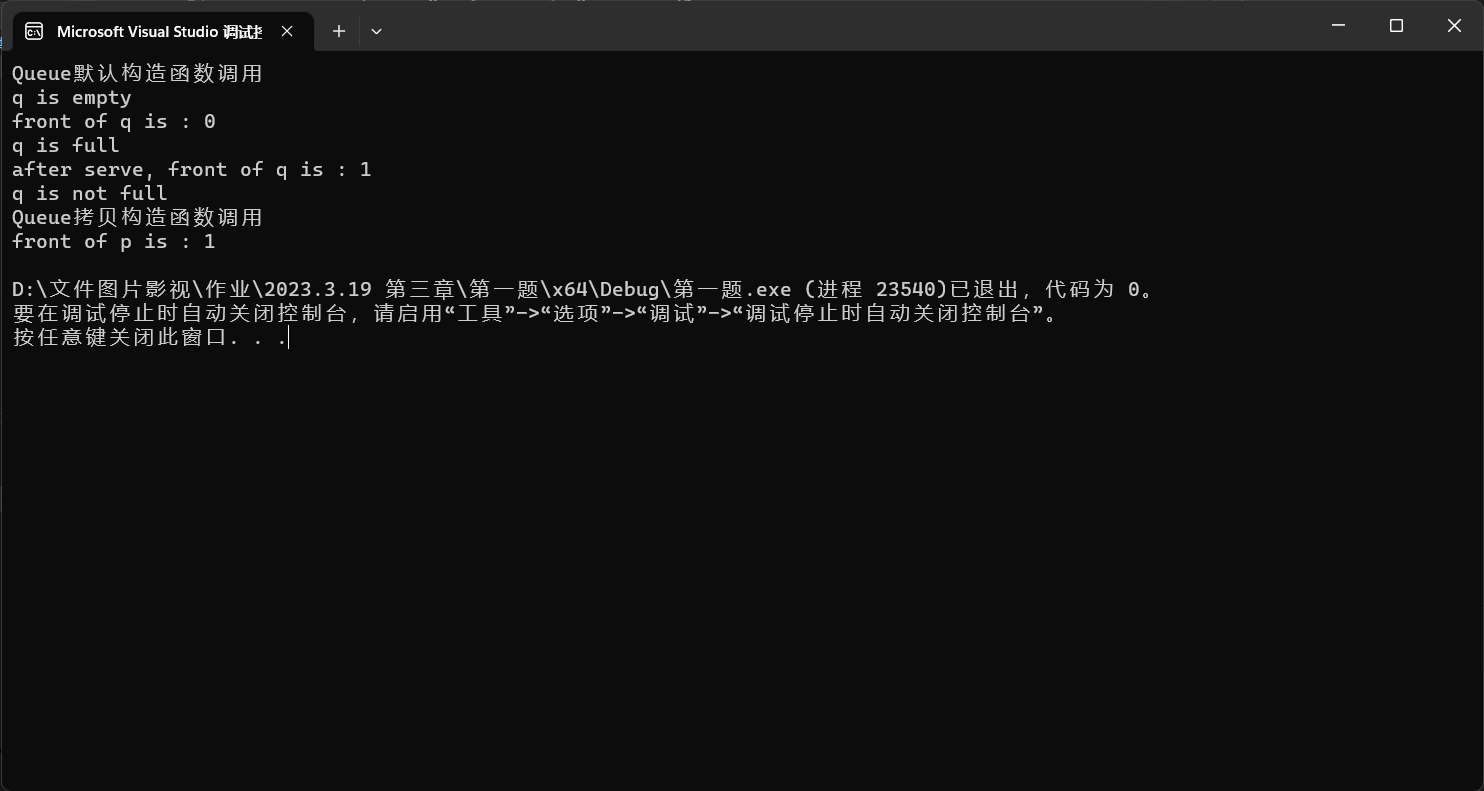
int main()

{

test();

return 0;

}



第二题

test.cpp

#include"Queue.hpp"

int main()

{

Queue<int> queue;

for (int i = 0; i < 50; i++)

{

queue.append(i);

}

cout << "count of queue : " << queue.length() << endl;

int length,head,rear;

head = queue.getfrontsign();

rear = queue.getrearsign();

length = (head < rear) ? rear - head : rear - head + maxlen;

cout << "from front to rear of queue : " << length << endl;

return 0;

}

Queue.hpp

#pragma once

#pragma once

#include<iostream>

using namespace std;

const int maxlen = 100;

template<class T>

class Queue //循环数组队列

{

public:

Queue();

Queue(Queue<T>& Q);

int length()const;

int getfrontsign()const;

int getrearsign()const;

bool empty()const;

bool full()const; //原理：为区别队空，留一个空(rear + 1) % maxlen == front

bool get\_front(T& x)const;

bool append(const T x);

bool serve(); //出队

private:

int count;

int front, rear; //队头，队尾,头指针不含元素

T data[maxlen];

};

template<class T>

Queue<T>::Queue()

{

count = 0;

front = rear = 0;

}

template<class T>

Queue<T>::Queue(Queue<T>& Q)

{

count = Q.count;

front = Q.count;

rear = Q.rear;

for (int i = front + 1; i % maxlen == rear; i++)

{

data[i] = Q.data[i];

}

}

template<class T>

int Queue<T>::length()const

{

return count;

}

template<class T>

int Queue<T>::getfrontsign()const

{

return front;

}

template<class T>

int Queue<T>::getrearsign()const

{

return rear;

}

template<class T>

bool Queue<T>::empty()const

{

//if (count == 0)

// return true;

//return false;

return front == rear;

}

template<class T>

bool Queue<T>::full()const

{

//if (count == maxlen - 1)

// return true;

//return false;

return (rear + 1) % maxlen == front;

}

template<class T>

bool Queue<T>::get\_front(T& x)const

{

if (empty())

return false;

x = data[(front + 1) % maxlen];

return true;

}

template<class T>

bool Queue<T>::append(const T x)

{

if (full())

return false;

rear = (rear + 1) % maxlen;

data[rear] = x;

count++;

return true;

}

template<class T>

bool Queue<T>::serve() //出队

{

if (empty())

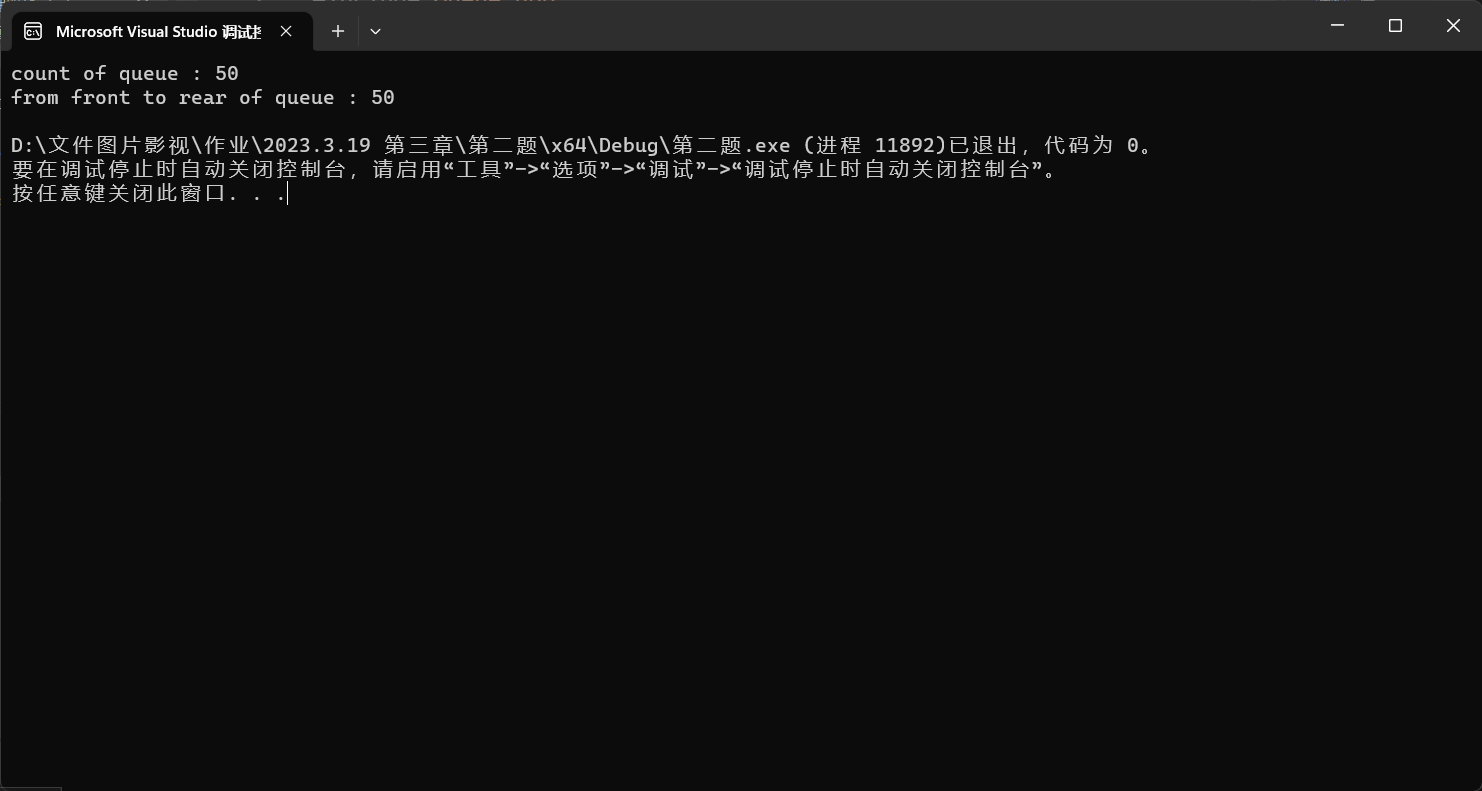
return false;

front = (front + 1) % maxlen;

count--;

return false;

}



第三题

class Queue //循环数组队列

{

public:

Queue();

Queue(Queue<T>& Q);

bool empty()const;

bool full()const; //原理：为区别队空，留一个空(rear + 1) % maxlen == front

bool get\_front(T& x)const;

bool append(const T x);

bool serve(); //出队

private:

int count;

int front, rear; //队头，队尾,头指针不含元素

T data[maxlen];

};

template<class T>

bool Queue<T>::full()const

{

//if (count == maxlen - 1)

// return true;

//return false;

return (rear + 1) % maxlen == front;

}

template<class T>

bool Queue<T>::get\_front(T& x)const

{

if (empty())

return false;

x = data[(front + 1) % maxlen];

return true;

}