《面向对象程序设计实验》任务书

# 实验二 面向对象的整型队列编程

**一、实验目的与要求**

1．熟悉类的定义方法，从实验一面向过程编程转向面向对象编程；

2．熟悉面向对象程序设计方法；

3．理解面向过程与面向对象的区别；

4．实现队列类的构造、析构、入队、出对等函数；

5．尝试基本的程序改进和优化手段并进行实验对比分析。

**二、实验内容**

将实验一通过面向过程程序设计实现的整型队列改造成如下要求的面向对象程序设计实现的整型队列。其中整型循环队列类QUEUE类的定义如下：

class QUEUE {

int\* const elems; //elems申请内存用于存放队列的元素

const int max; //elems申请的最大元素个数max

int head; //队列头head

int tail; //队列尾tail

public:

QUEUE(int m); //初始化队列，最多申请m个元素

QUEUE(const QUEUE& q); //用q深拷贝初始化队列

QUEUE(QUEUE&& q) noexcept; //用q移动初始化队列

int queSize () const; //返回队列申请的最大元素个数max

int queNumber () const; //返回队列的实际元素个数

QUEUE& queEnter(int e); //将e加入队列尾部，并返回当前队列

QUEUE& queEnter(short n, ...); //批量加入元素到队列并返回当前队列

QUEUE& queLeave(int& e); //从队首移出元素到e，并返回当前队列

QUEUE& queLeave(int& n, int\* buf); //批量移出n个元素到buf，并返回p

QUEUE& operator=(const QUEUE& q); //深拷贝赋值并返回被赋值队列

QUEUE& operator=(QUEUE&& q) noexcept; //移动赋值并返回被赋值队列

QUEUE& queCat(const QUEUE& q); //将q拼接到p队列末尾并返回p

void quePrint(char \*s) const; //打印队列至s并返回s

void queClear(); //清空并重置p指向的队列

~QUEUE(); //销毁当前队列

};

**需要注意是事项：**

1．**请不要自行添加定义任何新的数据成员和成员函数。**

2．队列应实现为循环队列，当队尾指针tail快要追上队首指针head时，即如果满足(tail+1)%max == head，则表示表示队列已满，故队列最多存放max-1个元素；当head == tail时则表示队列为空。

3. QUEUE(int m); 要求：m <= 0时应报错并输出错误提示信息。

4．待实现的成员函数的实现要求参照实验一对应的函数功能要求。

5．编程时应采用VS2019开发，并将其编译模式设置为x86模式。

6．请先实现完成队列操作的所有成员函数，然后编写一个main函数对这些函数进行测试。

1. 部分新增功能对应的成员函数的要求如下：
2. QUEUE(QUEUE&& q) noexcept; 要求：移动构造，在用已经存在的对象q移动构造新对象时，新对象使用对象q为elems分配的内存，并且新对象的max、head、tail应设置成和对象q的对应值相同；然后对象q的elems设置为空指针以表示内存被移走，同时其max、head、tail均应设置为0。
3. QUEUE& operator=(QUEUE&& q) noexcept; 要求：移动赋值，在用已经存在的对象q移动赋值给等号左边的对象时，等号左边的对象若为elems分配了内存，则应先释放内存以避免内存泄漏。等号左边的对象接受使用q为elems分配的内存，并且等号左边的对象的max、head、tail应设置成和对象q的对应值相同；对象q的elems然后设置为空指针以表示内存被移走，同时其max、head、tail均应置为0。
4. QUEUE& queEnter(short n, …); 要求：使用可变参数提供n个整型元素，如果队列剩余空间容纳不下，则保持执行前的队列状态不变，报错并输出错误提示信息；如果n<=0，则报错并输出错误提示信息。
5. QUEUE& queLeave(int& n, int\* buf); 要求：从队列移出不多于n个元素到buf，并返回p和实际移出的元素个数n。如果n<=0，则应报错并输出错误提示信息；如果buf为nullptr应报错并输出错误提示信息；如果队列为空，则返回的元素个数为0；如果队列元素个数不大于n，则移出所有元素并返回实际移出的元素个数。移出的元素按照先后顺序依次写入buf末尾。
6. QUEUE& queCat(const QUEUE& q); 要求：如果p指向的队列的剩余空间容纳不下q的所有元素，则按照至少可容纳两个队列元素个数之和的大小重新为p指向的队列分配内存，并实现两个队列的拼接。执行前后q队列保持不变，若p指向的队列重新分配了内存，则其head从0开始，并且须释放重新分配之前的内存以免泄漏。