《面向对象程序设计实验》任务书

# 实验四 面向对象的整型栈编程（异常处理与运算符重载）

**一、实验目的与要求**

1．掌握运算符重载的方法；

2．掌握异常处理的方法；

2．学习掌握常用的容器的使用方法。

**二、实验内容**

在实验三的基础上，将主要成员函数以运算符重载的形式改造实现。修改后的类定义如下：

#include <list>

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);

QUEUE(const QUEUE& q);

QUEUE(QUEUE&& q) noexcept;

virtual int size () const noexcept;

virtual operator int() const noexcept; //返回队列的实际元素个数

virtual QUEUE& operator<<(int e); //加入单个元素e至队列

virtual QUEUE& operator<<(std::list<int>& s); //批量加入元素至队列

virtual QUEUE& operator>>(int& e); //移出单个元素到e

virtual QUEUE& operator>>(std::list<int>& s); //批量移出元素到集合s

virtual QUEUE& operator=(const QUEUE& q);

virtual QUEUE& operator=(QUEUE&& q) noexcept;

virtual void print(char \*s) const;

virtual void clear()noexcept;

virtual ~QUEUE()noexcept;

};

class STACK : public QUEUE {

QUEUE q;

public:

STACK(int m);

STACK(const STACK& s);

STACK(STACK&& s) noexcept;

int size()const noexcept;

operator int() const noexcept; //返回栈的实际元素个数

STACK& operator<<(int e); //将e入栈，并返回当前栈

STACK & operator<<(std::list<int>& s); //将元素批量入栈

STACK& operator>>(int& e); //出栈到e，并返回当前栈

STACK & operator>>(std::list<int>& s); //批量出栈

STACK& operator=(const STACK& s);

STACK& operator=(STACK&& s) noexcept;

void print(char \*s)const;

void clear()noexcept;

~STACK()noexcept;

};

编程时应采用VS2019开发，并将其编译模式设置为x86模式。

**请不要添加定义任何新的数据成员和成员函数。**

注意：

（1）virtual QUEUE& operator<<(std::list<int>& s); 批量入队列时，如果容器size是0，则不做任何操作直接返回。

（2）virtual QUEUE & operator>>(std::list<int>& s); 批量出队列时，如果容器size是0，则固定最多从队列中移除5个元素；否则最多移出不超过size个元素。移出的元素应从容器起始位置开始放置。

（3）STACK & operator<<(std::list<int>& s); 批量入栈时，如果容器size是0，则不做任何操作直接返回。

（4）STACK & operator>>(std::list<int>& s); 批量出栈时，如果容器size是0，则固定最多从栈中移除5个元素；否则最多移出不超过size个元素。移出的元素应从容器起始位置开始放置。

（5）实验三中所有报错并给出提示信息的场景应转换成抛出异常，在main函数中的测试程序里面应能捕获相应的异常并加以处理，确保程序不会因异常而非正常终止。