Индивидуальное задание № 4

Тема: Наследование

Разработать иерархию классов:

Абстрактный класс очередь

Очередь на списках ***Очередь на массивах(кольцевая)***

Заголовочный файл приведен ниже. Необходимо реализовать все указанные методы. При необходимости можно добавлять другие методы. Продемонстрировать работоспособность ваших классов для статических и динамических объектов.

/*Queue.h*/

```
#ifndef __Queue_defined__
#define __Queue_defined__

#include <iostream>
using namespace std;

typedef int InfoType;

class QException: public exception {
public:
     QException(const char* message): exception(message) {}
     QException(const QException &right): exception(right) {}
};

class Queue {
private:
     virtual void Erase()=0 {};
     virtual void Clone(const Queue&)=0 {};
     /*методы с пустыми телами, будут вызваны в базовом классе при
       реализации конструктора копирования, деструктора и оператора
       присваивания*/
public:
     Queue() {}
     Queue(const Queue&);
     virtual ~Queue();
     virtual Queue& operator = (const Queue&);

     /*далее идут абстрактные методы (чисто виртуальные)
       для базового класса */
```

```
     virtual void Push(InfoType AInfo) = 0;
     virtual bool Pop() = 0;
     virtual InfoType GetFirst() const = 0;
     virtual bool IsEmpty()const = 0;

     virtual unsigned GetSize() const = 0;
     virtual InfoType& operator [] (unsigned) = 0;
     virtual const InfoType& GetByIndex(unsigned) const = 0;
     virtual void Browse(void ItemWork(InfoType)) const = 0;
     virtual void Browse(void ItemWork(InfoType&)) = 0;
};

class LQueue: public Queue {
private:
     struct QItem {
          InfoType info;
          QItem* next;
          QItem(InfoType Ainfo): info(Ainfo), next(NULL) {}
     };
     QItem *front, *rear;
     unsigned size;

     virtual void Erase();
     virtual void Clone(const Queue&);
public:
     LQueue(): front(NULL), rear(NULL), size(0) {};
     LQueue(const LQueue&);
     //virtual ~LQueue(); // только сообщение о вызове

     virtual void Push(InfoType AInfo);
     virtual bool Pop();
     virtual InfoType GetFirst() const;
     virtual bool IsEmpty()const;

     virtual unsigned GetSize() const;
     virtual InfoType& operator [] (unsigned);
     virtual const InfoType& GetByIndex(unsigned) const;
     virtual void Browse(void ItemWork(InfoType)) const;
     virtual void Browse(void ItemWork(InfoType&));
};

class CQueue: public Queue {
private:
     InfoType* Arr;
     unsigned first, last, capacity;
     void Erase();
     void Clone(const Queue&);
public:
     CQueue(int cpty = 100);
```

```
    CQueue(const CQueue&);
    //virtual ~CQueue(); // только сообщение о вызове

    virtual void Push(InfoType AInfo);
    virtual bool Pop();
    virtual InfoType GetFirst() const;
    virtual bool IsEmpty()const;

    virtual unsigned GetSize() const;
    virtual InfoType& operator [] (unsigned);
    virtual const InfoType& GetByIndex(unsigned) const;
    virtual void Browse(void ItemWork(InfoType)) const;
    virtual void Browse(void ItemWork(InfoType&));
};

#endif
```