```
/*
 * Instituto Superior de Formación Técnica 151
 * Tecnicatura Superior en Análisis de Sistemas
 *
 * Programación II
 *
 * Queue Project ( without parametrization )
 * Queue Variant : circular
 *
 * Version:  Revision 1.0  (23/8/2015)
 *           Revision 1.1  (24/8/2015)
 *           Revision 1.2  (31/8/2015)
 *
 * Grupo:
 *
 * Santiago, Matías Gastón
 * Molina Burgos, Álvaro
 * Mato, Santiago
 * Sosa, Luis
 *
 *
 * File:    CircularQueue.hpp (TDC)
 *
 */


#ifndef QUEUE HPP
#define QUEUE HPP

#include "IQueue.hpp"
#include <iostream>
#include <assert.h>
#include <malloc.h>
#include <string.h>

using namespace std;

class Queue : public IQueue
{
    private:

    enum { MAX ITEMS = 256, MAX HEAD=MAX ITEMS, SIZE DEFAULT = 32 };


    int head pos;

    int counter;
    int itemsize;

    char* block;

    int size() const
    {
        return itemsize*MAX ITEMS;
    };

    bool haveSpace() const
    {
        return ( counter < MAX ITEMS )? true : false;
    };

    int& validHead(int& counterValue )
    {
        if ( head pos == MAX ITEMS )
        {
            head pos = 0;
        };

        counterValue--;
        return head pos;
    };


```

```
74
75      int indexToSize(int index) const
76      {
77          return itemsize*index;
78      };
79
80      bool isValidSize( const Queue& queue ) const
81      {
82          return ( queue.size() <= size() )? true : false;
83      };
84
85      public:
86
87      virtual void add( void* item )
88      {
89          if ( haveSpace() )
90          memcpy( &block[ indexToSize(counter++) ], item, itemsize );
91      };
92
93      virtual bool isEmpty()
94      {
95          return ( counter == 0 )? true : false;
96      };
97
98      virtual bool isFull()
99      {
100         return ( counter == MAX ITEMS )? true : false;
101     };
102
103     virtual void* pop()
104     {
105         return &block[ indexToSize( validHead(counter)++ ) ];
106     };
107
108     //assignment operator
109     Queue& operator=( const Queue& queue )
110     {
111         if ( isValidSize(queue) )
112         {
113             head pos = queue.head pos;
114             counter = queue.counter;
115             itemsize = queue.itemsize;
116             memcpy( block, queue.block, queue.size() );
117         };
118
119         return *this;
120     };
121
122     //copy constructor
123     Queue( const Queue& queue ) :
124
125         head pos(queue.head pos),
126         counter(queue.counter),
127         itemsize(queue.itemsize),
128
129         block( (char*)memcpy( malloc( size() ), queue.block, size() ))
130
131     {};
132
133     //void constructor (with default parameter)
134     Queue( size_t typesize = SIZE DEFAULT ) :
135
136         head pos(0),
137         counter(0),
138         itemsize(typesize),
139         block( (char*)malloc( size() ) )
140
141     {};
142
143
144
145
146
```

```
147         //destructor
148         ~Queue()
149         {
150             free( block );
151         };
152
153
154
155     };
156
157     #endif
158
159
```