

```

1  /*
2  * Instituto Superior de Formación Técnica 151
3  * Tecnicatura Superior en Análisis de Sistemas
4  *
5  * Programación II
6  *
7  * Queue Project ( without parametrization )
8  * Queue Variant : simple
9  *
10 * Version:  Refactoring 1 (23/8/2015)
11 *           Revision 1.1  (24/8/2015)
12 *
13 * Grupo:
14 *
15 * Santiago, Matías Gastón
16 * Molina Burgos, Álvaro
17 * Mato, Santiago
18 * Sosa, Luis
19 *
20 *
21 * File:      QueueGeneric.hpp (TDC)
22 *
23 */
24
25 #ifndef QUEUE_GENERIC_HPP
26 #define QUEUE_GENERIC_HPP
27
28 #include "IQueue.hpp"
29 #include <assert.h>
30 #include <malloc.h>
31 #include <string.h>
32
33
34 class Queue : public IQueue
35 {
36     private:
37
38     typedef char byte;
39
40     enum { FIRST = 1, MAX ITEMS = 256, TOTAL ITEMS=257, NO ITEMS=0, SIZE DEFAULT = 32 };
41
42     size_t size item;
43     int items;
44
45     byte* block;
46
47     int indexToSize(int index)
48     {
49         return size item*index;
50     };
51
52
53     public:
54
55     virtual bool add(void* item)
56     {
57         bool state = !isFull();
58
59         if ( state )
60         {
61             memcpy( &block[ indexToSize(++items) ], item, size item );
62         };
63
64         return state;
65     }
66
67
68

```

```

69     virtual bool isFull()
70     {
71         return ( items == MAX ITEMS )? true : false;
72     }
73
74     virtual bool isEmpty()
75     {
76         return ( items == NO ITEMS )? true : false;
77     }
78
79     virtual void* read()
80     {
81         assert( !isEmpty() );
82         return &block[ indexToSize(FIRST) ];
83     }
84
85     virtual void* pop()
86     {
87         assert( !isEmpty() );
88         memcpy( block, &block[indexToSize(FIRST)], size item*(--items+1) );
89         return block;
90     }
91
92     //assignment operator
93     Queue& operator=( const Queue& queue )
94     {
95         size item = queue.size item;
96         items = queue.items;
97         memcpy( block, queue.block, queue.size item*TOTAL ITEMS );
98
99         return *this;
100    };
101
102    //copy constructor
103    Queue( const Queue& queue ) :
104    {
105        size item(queue.size item),
106        items(queue.items),
107        block(
108            (byte*)memcpy( (byte*)malloc(size item*TOTAL ITEMS),
109                queue.block, queue.size item*TOTAL ITEMS )
110        )
111    }
112    {};
113
114    //void constructor (with default parameter)
115    Queue( size_t bytes = SIZE DEFAULT ) :
116    {
117        size item(bytes),
118        items(0),
119        block( (byte*)malloc(size item*TOTAL ITEMS) )
120    }
121    {};
122
123    ~Queue()
124    {
125        free( block );
126    };
127
128
129 };
130
131 #endif
132
133

```