INE5408-03208A|INE5609-03238B (20182) - Estruturas de Dados

NAVEGAÇÃO

Painel
- Página inicial do site
- Moodle UFSC
- Curso atual
  - INE5408-03208A|INE5609-03238B (20182)
    - Participantes
    - Emblemas
    - Geral
    - Tópico 1
    - Tópico 2
      - Vídeos de Simulações de Filas e Pilhas
      - Video Aula sobre Pilhas e Filas com Vetores
      - Implementacão de Pilha com vetor
      - Testes (Pilha)
      - Implementacão de Fila com vetor
        - Descrição
        - Enviar
        - Editar
        - Visualizar envios
      - Testes (Fila)
      - Lecture 3
    - Tópico 3
    - Tópico 4
- Meus cursos

ADMINISTRAÇÃO
- Administração do curso

Descrição | Enviar | Editar | Visualizar envios

Nota

Revisado em domingo, 19 Ago 2018, 16:15 por Atribuição automática de nota
Nota 100 / 100
Relatório de avaliação
[+]**Summary of tests**
Enviado em domingo, 19 Ago 2018, 16:15 (Baixar)

array_queue.h

```
1  //! Copyright 2018 Matheus Henrique Schaly
2
3  #ifndef STRUCTURES_ARRAY_QUEUE_H
4  #define STRUCTURES_ARRAY_QUEUE_H
5
6  #include <cstdint>  // std::size_t
7  #include <stdexcept>  // C++ Exceptions
8
9  namespace structures {
10
11 //! Static queue
12 template<typename T>
13 class ArrayQueue {
14  public:
15 //! Constructor
16 ArrayQueue();
17
18 //! Constructor with parameter
19 ArrayQueue(std::size_t max);
20
21 //! Destructor
22 ~ArrayQueue();
23
24 //! Insert an element to the the end of the queue
25 void enqueue(const T& data);
26
27 //! Remove an element from the start of the queue
28 T dequeue();
29
30 //! Get the element from the back of the queue
31 T& back();
32
33 //! Clear the queue
34 void clear();
35
36 //! Return the size of the queue
37 std::size_t size();
38
39 //! Return the storage capacity of the queue
40 std::size_t max_size();
41
42 //! Return True if empty, false otherwise
43 bool empty();
44
45 //! Return True if full, false otherwise
46 bool full();
47
48  private:
49 T* contents;
50 std::size_t size_;
51 std::size_t max_size_;
52 static const auto DEFAULT_SIZE = 10u;
53 };
54 }  // namespace structures
55
56 template<typename T>
57 structures::ArrayQueue<T>::ArrayQueue() {
58     ArrayQueue(DEFAULT_SIZE);
59 }
60
61 template<typename T>
62 structures::ArrayQueue<T>::ArrayQueue(std::size_t max) {
63     size_ = 0;
64     max_size_ = max;
65     contents = new T[max];
66 }
67
68 template<typename T>
69 structures::ArrayQueue<T>::~ArrayQueue() {
70     delete[] contents;
71 }
72
73 template<typename T>
74 void structures::ArrayQueue<T>::enqueue(const T& data) {
75     if (full()) {
76         throw std::out_of_range("A fila esta cheia!");
77     } else {
78         for (int i = 0; i < size_; i++) {
79             contents[size_ - i] = contents[size_ - i - 1];
80         }
81         size_++;
82         contents[0] = data;
83     }
84 }
85
86 template<typename T>
87 T structures::ArrayQueue<T>::dequeue() {
88     if (empty()) {
89         throw std::out_of_range("A fila esta vazia!");
90     } else {
91         size_--;
92         return contents[size_];
93     }
94 }
95
96 template<typename T>
97 T& structures::ArrayQueue<T>::back() {
98     if (empty()) {
99         throw std::out_of_range("A fila esta vazia!");
100    } else {
101        return contents[0];
102    }
103 }
104
105 template<typename T>
106 void structures::ArrayQueue<T>::clear() {
107     size_ = 0;
108 }
109
110 template<typename T>
111 std::size_t structures::ArrayQueue<T>::size() {
112     return size_;
113 }
114
115 template<typename T>
116 std::size_t structures::ArrayQueue<T>::max_size() {
```

```
117     return max_size_;
118 }
119
120 template<typename T>
121 bool structures::ArrayQueue<T>::empty() {
122     return (size_ == 0);
123 }
124
125 template<typename T>
126 bool structures::ArrayQueue<T>::full() {
127     return (size_ == max_size_);
128 }
129
130 #endif
131
```