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
    - Tópico 3
    - Tópico 4
    - Tópico 5
    - Tópico 6
    - Tópico 7
    - Tópico 8
    - Tópico 9
    - Tópico 10
      - Implementação de Pilha Encadeada
        - Descrição
        - Enviar
        - Editar
        - Visualizar envios
      - Testes (Pilha Encadeada)
      - Implementação de Fila Encadeada
      - Testes (Fila Encadeada)
      - Implementação de Lista Duplamente Encadeada
      - Testes (Lista Duplamente Encadeada)
      - Videoaula de Pilha e Fila Encadeada
      - Lecture 8 (rev. 04-set)
- Meus cursos

ADMINISTRAÇÃO

- Administração do curso

Nota

Revisado em domingo, 9 Set 2018, 16:04 por Atribuição automática de nota
Nota 100 / 100
Relatório de avaliação
[+]**Summary of tests**
Enviado em domingo, 9 Set 2018, 16:04 (Baixar)

linked_stack.h

```cpp
//! Copyright 2018 Matheus Henrique Schaly

#ifndef STRUCTURES_LINKED_STACK_H
#define STRUCTURES_LINKED_STACK_H

#include <cstdint>
#include <stdexcept>


namespace structures {

//! Dynamic stack implementation
template<typename T>
class LinkedStack {
 public:
    //! Constructor
    LinkedStack();

    //! Destructor
    ~LinkedStack();

    //! Clears the stack
    void clear();

    //! Inserts an element at the rightmost of the stack
    void push(const T& data);

    //! Removes an element from the rightmost of the stack
    T pop();

    //! Returns the data at the rightmost node of the stack
    T& top() const;

    //! Returns true if stack is empty and false otherwise
    bool empty() const;

    //! Returns the current size of the queue
    std::size_t size() const;

 private:
    class Node {
     public:
        //! Constructor with 1 parameter
        explicit Node(const T& data):
            data_{data}
        {}

        //! Constructor with 2 parameters
        Node(const T& data, Node* next):
            data_{data},
            next_{next}
        {}

        //! Info's getter
        T& data() {
            return data_;
        }

        //! Info's constant getter
        const T& data() const {
            return data_;
        }

        //! Next's getter
        Node* next() {
            return next_;
        }

        //! Next's constant getter
        const Node* next() const {
            return next_;
        }

        //! Next's setter
        void next(Node* next) {
            next_ = next;
        }

     private:
        //! Node's data
        T data_;

        //! Node's next node
        Node* next_;
    };

    //! Stack's rightmost node
    Node* top_{nullptr};

    //! Stack's current size
    std::size_t size_{0u};
};

}  // namespace structures

template<typename T>
structures::LinkedStack<T>::LinkedStack() {}

template<typename T>
structures::LinkedStack<T>::~LinkedStack() {
    clear();
}

template<typename T>
void structures::LinkedStack<T>::clear() {
    while (!empty()) {
        pop();
    }
}

template<typename T>
void structures::LinkedStack<T>::push(const T& data) {
    Node* node = new Node(data, nullptr);
    if (node == nullptr) {
        throw std::out_of_range("A pilha esta cheia.");
    }
    if (empty()) {
        top_ = node;
    } else {
        node -> next(top_);
        top_ = node;
    }
    size_++;
}

template<typename T>
T structures::LinkedStack<T>::pop() {
    if (empty()) {
        throw std::out_of_range("A pilha esta vazia.");
    }
    Node* node = top_;
    T deleted_data = node -> data();
    top_ = top_ -> next();
    delete node;
    size_--;
    return deleted_data;
}

template<typename T>
T& structures::LinkedStack<T>::top() const {
    if (empty()) {
        throw std::out_of_range("A pilha esta vazia.");
    }
    return top_ -> data();
}

template<typename T>
bool structures::LinkedStack<T>::empty() const {
    return size_ == 0;
}

template<typename T>
std::size_t structures::LinkedStack<T>::size() const {
    return size_;
}

#endif

```

VPL 3.1.5