Implementação 1 - Grafos

Samuel Correia , Vínicius Ferrer , Daniel Salgado , Arthur Martinho

11/08/2024

1 Lista implementada em C++

```cpp
#include <iostream>
#include <initializer_list>

struct Node {
    int data;
    Node* next;

    Node(int value) : data(value), next(nullptr) {}
};

class LinkedList {
private:
    Node* head;

public:
    // Construtor
    LinkedList() : head(nullptr) {}

    ~LinkedList() { // Destruidor
        Node* current = head;
        while (current != nullptr) {
            Node* nextNode = current->next;
            delete current;
            current = nextNode;
        }
    }

    // Inserir m ltiplos elementos na cabe a da lista
    void inseretopo(std::initializer_list<int> values) {
        for (int value : values) {
            Node* newNode = new Node(value);
            newNode->next = head;
            head = newNode;
        }
    }

    // Inserir m ltiplos elementos no final da lista
    void inserefim(std::initializer_list<int> values) {
        for (int value : values) {
            Node* newNode = new Node(value);
            if (head == nullptr) {
                head = newNode;
            } else {
                Node* current = head;
                while (current->next != nullptr) {
                    current = current->next;
                }
                current->next = newNode;
            }
        }
    }

    // Remover m ltiplos elementos pela cabe a da lista
    void removerTopo(int count) {
```

```
        for (int i = 0; i < count && head != nullptr; ++i) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
    }

    // Remover m ltiplos elementos do final da lista
    void removerFim(int count) {
        while (count-- > 0 && head != nullptr) {
            if (head->next == nullptr) {
                delete head;
                head = nullptr;
            } else {
                Node* current = head;
                while (current->next->next != nullptr) {
                    current = current->next;
                }
                delete current->next;
                current->next = nullptr;
            }
        }
    }

    // M todo para verificar se a lista cont m um determinado valor
    bool contem(int value) const {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == value) {
                return true;
            }
            current = current->next;
        }
        return false;
    }

    void printLista() const {
        Node* current = head;
        while (current != nullptr) {
            std::cout << current->data << " ";
            current = current->next;
        }
        std::cout << std::endl;
    }
};

int main() {
    LinkedList list;

    list.inseretopo({10, 20, 30});
    list.inserefim({40, 50, 60});


    std::cout << "Lista encadeada: ";
    list.printLista();

    list.removerTopo(2);
    std::cout << "Lista ap s remover 2 elementos do topo: ";
    list.printLista();

    list.removerFim(2);
    std::cout << "Lista ap s remover 2 elementos do fim: ";
    list.printLista();

    // Verificar se a lista cont m um valor
    int valorProcurado = 40;
    if (list.contem(valorProcurado)) {
        std::cout << "A lista cont m o valor " << valorProcurado << "." << std::endl;
    } else {
        std::cout << "A lista n o cont m o valor " << valorProcurado << "." << std::
    endl;
```

```
    }

    return 0;
}
```

Listing 1: Exemplo C++

1.1 Como Funciona

Estrutura Node: Contém um valor (data) e um ponteiro para o próximo nó (next).
O construtor inicializa o nó com um valor e define o ponteiro next como nullptr(Null).

Classe LinkedList: Mantém um ponteiro para o início da lista (head). O destruidor libera a memória alocada para a lista, evitando vazamentos de memória.

Funções:
inseretopo: Adiciona N novos nós no início da lista.
inserefim: Adiciona N novos nós no final da lista.
removerTopo: Remove K nós do inicio da lista.
removerFim: Remove K nós do inicio da lista.
printList: imprime todos os valores na lista.
contem checa se um determinado valor está presente na lsita

Função main: Demonstra o uso das funções da lista encadeada inserindo alguns valores e imprimindo a lista resultante.

1.2 Teste do Codigo

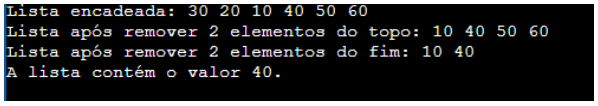

Figure 1: Teste fila

2 Pilha implementada em C++

```

#include <iostream>
#include <initializer_list>

// Estrutura do  n   da pilha
struct Node {
    int data;         // Dados armazenados no  n
    Node* next;       // Ponteiro para o pr ximo  n

    // Construtor para inicializar o  n   com um valor
    Node(int value) : data(value), next(nullptr) {}
};

// Classe para a pilha
class Stack {
private:
    Node* top;       // Ponteiro para o topo da pilha

public:
    // Construtor
    Stack() : top(nullptr) {}

```

```
    // Destruidor
    ~Stack() {
        // Libera toda a mem ria alocada para a pilha
        while (!isEmpty()) {
            remover();
        }
    }

    // Fun  o para adicionar um elemento na pilha
    void insere(int value) {
        Node* newNode = new Node(value);
        newNode->next = top;
        top = newNode;
    }

    // Fun  o para adicionar n elementos na pilha
    void insereN(std::initializer_list<int> values) {
        for (int value : values) {
            insere(value);
        }
    }

    // Fun  o para remover o elemento do topo da pilha
    void remover() {
        if (isEmpty()) {
            std::cerr << "Pilha vazia. N o    poss vel remover elementos." << std::
endl;
            return;
        }
        Node* temp = top;
        top = top->next;
        delete temp;
    }

    // Fun  o para remover k elementos do topo da pilha
    void removerK(int count) {
        for (int i = 0; i < count; ++i) {
            if (!isEmpty()) {
                remover();
            } else {
                // checagem se a pilha n o ficou vazia
                std::cerr << "Pilha vazia ap s remover " << i << " elementos." << std
::endl;
                break;
            }
        }
    }

    // Fun  o para obter o elemento do topo da pilha
    int topo() const {
        if (isEmpty()) {
            std::cerr << "Pilha vazia. N o    poss vel acessar o elemento do topo."
<< std::endl;
            return -1; // Valor de erro
        }
        return top->data;
    }

    // Fun  o para verificar se a pilha est  vazia
    bool isEmpty() const {
        return top == nullptr;
    }

    // Fun  o para imprimir todos os elementos da pilha
    void printPilha() const {
        Node* current = top;
        while (current != nullptr) {
            std::cout << current->data << " ";
            current = current->next;
        }
        std::cout << std::endl;
```

```
    }

    // Fun   o para verificar se a pilha cont m um elemento espec fico
    bool contem(int value) const {
        Node* current = top;
        while (current != nullptr) {
            if (current->data == value) {
                return true;
            }
            current = current->next;
        }
        return false;
    }
};

int main() {
    Stack stack;

    // Adicionando m ltiplos elementos    pilha
    stack.insereN({10, 20, 30, 40});
    stack.insere(50);

    std::cout << "Pilha atual: ";
    stack.printPilha();

    std::cout << "Topo da pilha: " << stack.topo() << std::endl;

    // Removendo m ltiplos elementos da pilha
    stack.removerK(2);
    stack.remover();
    std::cout << "Pilha ap s removerK e remover: ";
    stack.printPilha();

    std::cout << "Topo da pilha ap s removerK e remover: " << stack.topo() << std::
    endl;

    // Verificando se a pilha cont m determinados elementos
    int elemento = 20;
    if (stack.contem(elemento)) {
        std::cout << "A pilha cont m o elemento " << elemento << "." << std::endl;
    } else {
        std::cout << "A pilha n o cont m o elemento " << elemento << "." << std::
    endl;
    }

    elemento = 50;
    if (stack.contem(elemento)) {
        std::cout << "A pilha cont m o elemento " << elemento << "." << std::endl;
    } else {
        std::cout << "A pilha n o cont m o elemento " << elemento << "." << std::
    endl;
    }

    return 0;
}
```

Listing 2: Exemplo C++

2.1 Como Funciona

Estrutura Node: Contém um valor (data) e um ponteiro para o próximo nó (next). O construtor inicializa o nó com um valor e define o ponteiro next como nullptr.

Classe Stack: Mantém um ponteiro para o topo da pilha (top).
O destruidor libera a memória alocada para a pilha, removendo todos os nós.

Comandos:
insere: Adiciona um novo nó no topo da pilha.

remover: Remove o nó do topo da pilha e libera a memória associada.
insereK: Adciona n elementos no topo da pilha com base na ordem de digitação.
removerK: Remove k elementos do topo da fila. topo retorna o valor do topo da pilha sem removê-lo. Retorna -1 se a pilha estiver vazia.
isEmpty: Verifica se a pilha está vazia.
printPilha: Imprime todos os elementos da pilha, do topo para a base.
contem: Verifica se um elemento especifico está presente na pilha

Função main: Demonstra o uso das funções da pilha, incluindo insere, remover,insereN,removerK, topo e printPilha

2.2 Teste do Codigo

```
Pilha atual: 50 40 30 20 10
Topo da pilha: 50
Pilha após removerK e remover: 20 10
Topo da pilha após removerK e remover: 20
A pilha contém o elemento 20.
A pilha não contém o elemento 50.
```

Figure 2: Teste pilha

3 Fila implementada em C++

```
1
2  #include <iostream>
3  #include <initializer_list>
4
5  // Estrutura do  n   da fila
6  struct Node {
7      int data;         // Dados armazenados no  n
8      Node* next;       // Ponteiro para o pr ximo  n
9
10     // Construtor para inicializar o  n   com um valor
11     Node(int value) : data(value), next(nullptr) {}
12 };
13
14 // Classe para a fila
15 class Queue {
16 private:
17     Node* front;     // Ponteiro para o in cio da fila
18     Node* rear;      // Ponteiro para o final da fila
19
20 public:
21     // Construtor
22     Queue() : front(nullptr), rear(nullptr) {}
23
24     // Destruidor
25     ~Queue() {
26         // Libera toda a mem ria alocada para a fila
27         while (!isEmpty()) {
28             remover();
29         }
30     }
31
32     // Fun   o para adicionar um elemento na fila
33     void insere(int value) {
34         Node* newNode = new Node(value);
35         if (isEmpty()) {
36             front = rear = newNode;
37         } else {
38             rear->next = newNode;
39             rear = newNode;
40         }
41     }
42
43     // Fun   o para adicionar m ltiplos elementos na fila
44     void insereN(std::initializer_list<int> values) {
45         for (int value : values) {
46             insere(value);
47         }
48     }
49
50     // Fun   o para remover o elemento do in cio da fila
51     void remover() {
52         if (isEmpty()) {
53             std::cerr << "Fila vazia. N o    poss vel remover elementos." << std::
    endl;
54             return;
55         }
56         Node* temp = front;
57         front = front->next;
58         if (front == nullptr) {
59             rear = nullptr; // Se a fila ficar vazia, rear tamb m deve ser nullptr
60         }
61         delete temp;
62     }
63
64     // Fun   o para remover m ltiplos elementos do in cio da fila
65     void removerK(int count) {
66         for (int i = 0; i < count; ++i) {
67             if (!isEmpty()) {
68                 remover();
```

```
            } else {
                std::cerr << "Fila vazia ap s remover " << i << " elementos." << std
    ::endl;
                break;
            }
        }
    }

    // Fun   o para verificar se a fila est  vazia
    bool isEmpty() const {
        return front == nullptr;
    }

    // Fun   o para verificar se um elemento est  na fila
    bool contem(int value) const {
        Node* current = front;
        while (current != nullptr) {
            if (current->data == value) {
                return true;
            }
            current = current->next;
        }
        return false;
    }

    // Fun   o para imprimir todos os elementos da fila
    void printFila() const {
        Node* current = front;
        while (current != nullptr) {
            std::cout << current->data << " ";
            current = current->next;
        }
        std::cout << std::endl;
    }
};

int main() {
    Queue fila;

    // Adicionando m ltiplos elementos    fila
    fila.insereN({10, 20, 30, 40});
    fila.insere(50);

    std::cout << "Fila atual: ";
    fila.printFila();

    // Remover m ltiplos elementos da fila
    fila.removerK(2);
    fila.remover();
    std::cout << "Fila ap s removerK e remover: ";
    fila.printFila();

    return 0;
}
```

Listing 3: Exemplo C++

3.1 Como Funciona

Estrutura Node: Contém um valor (data) e um ponteiro para o próximo nó (next). O construtor inicializa o nó com um valor e define o ponteiro next como nullptr.

Classe Queue: Mantém um ponteiro para o inicio e fim da fila (front e rear). O construtor,inicia a estrutura definindo o inicio e o fim como nulos
O destruidor libera a memória alocada para a pilha, removendo todos os nós.

Comandos:
insere: Adiciona um novo no fim da fila.

remover: Remove um nó do inicio da fila
insereN: Adciona n elementos no fim da fila.
removerK: Remove k elementos inicio da fila.
contem: Verifica se um elemento está presente na fila
isEmpty: Verifica se a fila está vazia.
printFila: Imprime todos os elementos da fila

Função main: Demonstra o uso das funções da fila, incluindo insere, remover,insereN,removerK,contem e printFila

3.2 Teste do Codigo

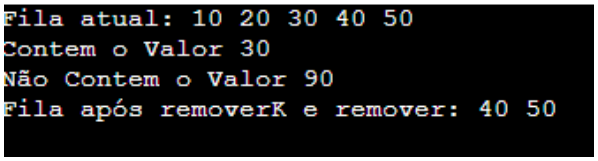

Figure 3: Teste fila

4 Matriz implementada em C++

```cpp
#include <iostream>

struct Node {
    int data;
    Node* next;

    Node(int value) : data(value), next(nullptr) {}
};

class Matriz {
private:
    Node* head;  // Ponteiro para o primeiro  n   da lista
    Node* ultimo;  // Ponteiro para o  ltimo   n   da lista

public:
    // Construtor
    Matriz() : head(nullptr), ultimo(nullptr) {}

    // Destrutor para liberar mem ria
    ~Matriz() {
        while (head != nullptr) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
    }

    // Inserir m ltiplos elementos no topo da matriz
    void insereTopo(std::initializer_list<int> values) {
        for (int value : values) {
            Node* newNode = new Node(value);
            newNode->next = head;
            head = newNode;
            if (ultimo == nullptr) {
                ultimo = head;
            }
        }
    }

    // Inserir m ltiplos elementos no final da matriz
    void insereFim(std::initializer_list<int> values) {
        for (int value : values) {
            Node* newNode = new Node(value);
            if (ultimo == nullptr) {
                head = ultimo = newNode;
            } else {
                ultimo->next = newNode;
                ultimo = newNode;
            }
        }
    }

    // Remover m ltiplos elementos do topo da matriz
    void removeTopo(int count) {
        while (count-- > 0 && head != nullptr) {
            Node* temp = head;
            head = head->next;
            delete temp;
            if (head == nullptr) {
                ultimo = nullptr;
            }
        }
    }

    // Remover m ltiplos elementos do final da matriz
    void removeFim(int count) {
        while (count-- > 0 && head != nullptr) {
            if (head->next == nullptr) {
                delete head;
```

```
                head = ultimo = nullptr;
            } else {
                Node* current = head;
                while (current->next != ultimo) {
                    current = current->next;
                }
                delete ultimo;
                ultimo = current;
                ultimo->next = nullptr;
            }
        }
    }

    // M todo para verificar se a matriz cont m um determinado valor
    bool contem(int value) const {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == value) {
                return true;
            }
            current = current->next;
        }
        return false;
    }

    // Imprimir os elementos da matriz
    void printMatriz() const {
        Node* current = head;
        while (current != nullptr) {
            std::cout << current->data << " ";
            current = current->next;
        }
        std::cout << std::endl;
    }
};

int main() {
    Matriz matriz;

    matriz.insereTopo({10, 20, 30});
    matriz.insereFim({40, 50, 60});

    std::cout << "Matriz: ";
    matriz.printMatriz();

    matriz.removeTopo(2);
    std::cout << "Matriz ap s remover K elementos do topo: ";
    matriz.printMatriz();

    matriz.removeFim(2);
    std::cout << "Matriz ap s remover K elementos do fim: ";
    matriz.printMatriz();

    // Verificar se a matriz cont m um valor
    int valorProcurado = 40;
    if (matriz.contem(valorProcurado)) {
        std::cout << "A matriz cont m o valor " << valorProcurado << "." << std::endl
    ;
    } else {
        std::cout << "A matriz n o  cont m o valor " << valorProcurado << "." << std
    ::endl;
    }

    return 0;
}
```

Listing 4: Exemplo C++

4.1 Como Funciona

Estrutura Node: Contém um valor (data) e um ponteiro para o próximo nó (next). O construtor inicializa o nó com um valor e define o ponteiro next como nullptr.

Classe Matriz: Mantém um ponteiro para o inicio e fim da fila (head e ultimo). O construtor,inicia a estrutura definindo o inicio e o fim como nulos. O destruidor libera a memória alocada para a matriz, removendo todos os nós.

Comandos:
insereTopo: Adiciona um novo elemento no começo da matriz.
insereFim: Adiciona um novo elemento no final da matriz.
removeTopo: Remove um nó do inicio da matriz.
removeFim: Remove um nó do final da matriz.
contem: Verifica se um elemento está presente na matriz
printMatriz: Imprime todos os elementos da matriz

Função main: Demonstra o uso das funções da matriz, incluindo os tipos de inserção(insereTopo, insereFim, insereN), remoção(removeTopo, removeFim, removeK), contem e printMatriz

4.2 Teste do Codigo

```
Matriz: 30 20 10 40 50 60
Matriz após remover K elementos do topo: 10 40 50 60
Matriz após remover K elementos do fim: 10 40
A matriz contém o valor 40.
```

Figure 4: Teste Matriz