

Implementação 1 - Grafos

Samuel Correia , Vínicius Ferrer , Daniel Salgado , Arthur Martinho

11/08/2024

1 Lista implementada em C++

```
1 #include <iostream>
2 #include <initializer_list>
3
4 struct Node {
5     int data;
6     Node* next;
7
8     Node(int value) : data(value), next(nullptr) {}
9 };
10
11 class LinkedList {
12 private:
13     Node* head;
14
15 public:
16     // Construtor
17     LinkedList() : head(nullptr) {}
18
19     ~LinkedList() { // Destruidor
20         Node* current = head;
21         while (current != nullptr) {
22             Node* nextNode = current->next;
23             delete current;
24             current = nextNode;
25         }
26     }
27
28     // Inserir mltiplos elementos na cabe a da lista
29     void inseretopo(std::initializer_list<int> values) {
30         for (int value : values) {
31             Node* newNode = new Node(value);
32             newNode->next = head;
33             head = newNode;
34         }
35     }
36
37     // Inserir mltiplos elementos no final da lista
38     void inserefim(std::initializer_list<int> values) {
39         for (int value : values) {
40             Node* newNode = new Node(value);
41             if (head == nullptr) {
42                 head = newNode;
43             } else {
44                 Node* current = head;
45                 while (current->next != nullptr) {
46                     current = current->next;
47                 }
48                 current->next = newNode;
49             }
50         }
51     }
52
53     // Remover mltiplos elementos pela cabe a da lista
54     void removerTopo(int count) {
```

```

55     for (int i = 0; i < count && head != nullptr; ++i) {
56         Node* temp = head;
57         head = head->next;
58         delete temp;
59     }
60 }
61
62 // Remover m ltiplos elementos do final da lista
63 void removerFim(int count) {
64     while (count-- > 0 && head != nullptr) {
65         if (head->next == nullptr) {
66             delete head;
67             head = nullptr;
68         } else {
69             Node* current = head;
70             while (current->next->next != nullptr) {
71                 current = current->next;
72             }
73             delete current->next;
74             current->next = nullptr;
75         }
76     }
77 }
78
79 // M todo para verificar se a lista cont m um determinado valor
80 bool contem(int value) const {
81     Node* current = head;
82     while (current != nullptr) {
83         if (current->data == value) {
84             return true;
85         }
86         current = current->next;
87     }
88     return false;
89 }
90
91 void printLista() const {
92     Node* current = head;
93     while (current != nullptr) {
94         std::cout << current->data << " ";
95         current = current->next;
96     }
97     std::cout << std::endl;
98 }
99 };
100
101 int main() {
102     LinkedList list;
103
104     list.inseretopo({10, 20, 30});
105     list.inserefim({40, 50, 60});
106
107
108     std::cout << "Lista encadeada: ";
109     list.printLista();
110
111     list.removeTopo(2);
112     std::cout << "Lista ap s remover 2 elementos do topo: ";
113     list.printLista();
114
115     list.removeFim(2);
116     std::cout << "Lista ap s remover 2 elementos do fim: ";
117     list.printLista();
118
119     // Verificar se a lista cont m um valor
120     int valorProcurado = 40;
121     if (list.contem(valorProcurado)) {
122         std::cout << "A lista cont m o valor " << valorProcurado << "." << std::endl;
123     } else {
124         std::cout << "A lista n o cont m o valor " << valorProcurado << "." << std::endl;
125     }

```

```

125     }
126
127     return 0;
128 }

```

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:

insetop: Adiciona N novos nós no início da lista.

inserefim: Adiciona N novos nós no final da lista.

removeTopo: Remove K nós do inicio da lista.

removeFim: Remove K nós do inicio da lista.

printList: imprime todos os valores na lista.

contem: verifica se um determinado valor está presente na lista

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

1.2 Teste do Código

```

Lista encadeada: 30 20 10 40 50 60
Lista após remover 2 elementos do topo: 10 40 50 60
Lista após remover 2 elementos do fim: 10 40
A lista contém o valor 40.

```

Figure 1: Teste fila

2 Pilha implementada em C++

```

1
2 #include <iostream>
3 #include <initializer_list>
4
5 // Estrutura do n da pilha
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 pilha
15 class Stack {
16 private:
17     Node* top;         // Ponteiro para o topo da pilha
18
19 public:
20     // Construtor
21     Stack() : top(nullptr) {}
22

```

```

23 // Destruidor
24 ~Stack() {
25     // Libera toda a mem ria alocada para a pilha
26     while (!isEmpty()) {
27         remover();
28     }
29 }
30
31 // Fun o para adicionar um elemento na pilha
32 void insere(int value) {
33     Node* newNode = new Node(value);
34     newNode->next = top;
35     top = newNode;
36 }
37
38 // Fun o para adicionar n elementos na pilha
39 void insereN(std::initializer_list<int> values) {
40     for (int value : values) {
41         insere(value);
42     }
43 }
44
45 // Fun o para remover o elemento do topo da pilha
46 void remover() {
47     if (isEmpty()) {
48         std::cerr << "Pilha vazia. N o poss vel remover elementos." << std::
endl;
49         return;
50     }
51     Node* temp = top;
52     top = top->next;
53     delete temp;
54 }
55
56 // Fun o para remover k elementos do topo da pilha
57 void removerK(int count) {
58     for (int i = 0; i < count; ++i) {
59         if (!isEmpty()) {
60             remover();
61         } else {
62             // checagem se a pilha n o ficou vazia
63             std::cerr << "Pilha vazia ap s remover " << i << " elementos." << std
::endl;
64             break;
65         }
66     }
67 }
68
69 // Fun o para obter o elemento do topo da pilha
70 int topo() const {
71     if (isEmpty()) {
72         std::cerr << "Pilha vazia. N o poss vel acessar o elemento do topo."
<< std::endl;
73         return -1; // Valor de erro
74     }
75     return top->data;
76 }
77
78 // Fun o para verificar se a pilha est vazia
79 bool isEmpty() const {
80     return top == nullptr;
81 }
82
83 // Fun o para imprimir todos os elementos da pilha
84 void printPilha() const {
85     Node* current = top;
86     while (current != nullptr) {
87         std::cout << current->data << " ";
88         current = current->next;
89     }
90     std::cout << std::endl;

```

```

91     }
92
93     // Função para verificar se a pilha contém um elemento específico
94     bool contem(int value) const {
95         Node* current = top;
96         while (current != nullptr) {
97             if (current->data == value) {
98                 return true;
99             }
100             current = current->next;
101         }
102         return false;
103     }
104 };
105
106 int main() {
107     Stack stack;
108
109     // Adicionando múltiplos elementos na pilha
110     stack.inserirN({10, 20, 30, 40});
111     stack.inserir(50);
112
113     std::cout << "Pilha atual: ";
114     stack.printPilha();
115
116     std::cout << "Topo da pilha: " << stack.topo() << std::endl;
117
118     // Removendo múltiplos elementos da pilha
119     stack.removerK(2);
120     stack.remover();
121     std::cout << "Pilha após removerK e remover: ";
122     stack.printPilha();
123
124     std::cout << "Topo da pilha após removerK e remover: " << stack.topo() << std::endl;
125
126     // Verificando se a pilha contém determinados elementos
127     int elemento = 20;
128     if (stack.contem(elemento)) {
129         std::cout << "A pilha contém o elemento " << elemento << "." << std::endl;
130     } else {
131         std::cout << "A pilha não contém o elemento " << elemento << "." << std::endl;
132     }
133
134     elemento = 50;
135     if (stack.contem(elemento)) {
136         std::cout << "A pilha contém o elemento " << elemento << "." << std::endl;
137     } else {
138         std::cout << "A pilha não contém o elemento " << elemento << "." << std::endl;
139     }
140
141     return 0;
142 }

```

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:

inserir: Adiciona um novo nó no topo da pilha.

remove: Remove o nó do topo da pilha e libera a memória associada.
insereK: Adiciona n elementos no topo da pilha com base na ordem de digitação.
removeK: 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 específico está presente na pilha

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

2.2 Teste do Código

```
Pilha atual: 50 40 30 20 10
Topo da pilha: 50
Pilha após removeK e remove: 20 10
Topo da pilha após removeK e remove: 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();
```

```

69         } else {
70             std::cerr << "Fila vazia ap s remover " << i << " elementos." << std
::endl;
71             break;
72         }
73     }
74 }
75
76 // Fun o para verificar se a fila est vazia
77 bool isEmpty() const {
78     return front == nullptr;
79 }
80
81 // Fun o para verificar se um elemento est na fila
82 bool contem(int value) const {
83     Node* current = front;
84     while (current != nullptr) {
85         if (current->data == value) {
86             return true;
87         }
88         current = current->next;
89     }
90     return false;
91 }
92
93 // Fun o para imprimir todos os elementos da fila
94 void printFila() const {
95     Node* current = front;
96     while (current != nullptr) {
97         std::cout << current->data << " ";
98         current = current->next;
99     }
100     std::cout << std::endl;
101 }
102 };
103
104 int main() {
105     Queue fila;
106
107     // Adicionando m ltiplos elementos fila
108     fila.inseraN({10, 20, 30, 40});
109     fila.insera(50);
110
111     std::cout << "Fila atual: ";
112     fila.printFila();
113
114     // Remover m ltiplos elementos da fila
115     fila.removeK(2);
116     fila.remove();
117     std::cout << "Fila ap s removeK e remove: ";
118     fila.printFila();
119
120     return 0;
121 }

```

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 início e fim da fila (front e rear). O construtor, inicia a estrutura definindo o início 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 início da fila
insereN: Adiciona n elementos no fim da fila.
removerK: Remove k elementos início 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 Código

```
Fila atual: 10 20 30 40 50  
Contem o Valor 30  
Não Contem o Valor 90  
Fila após removerK e remover: 40 50
```

Figure 3: Teste fila

4 Matriz implementada em C++

```
1 #include <iostream>
2
3 struct Node {
4     int data;
5     Node* next;
6
7     Node(int value) : data(value), next(nullptr) {}
8 };
9
10 class Matriz {
11 private:
12     Node* head; // Ponteiro para o primeiro n da lista
13     Node* ultimo; // Ponteiro para o ltimo n da lista
14
15 public:
16     // Construtor
17     Matriz() : head(nullptr), ultimo(nullptr) {}
18
19     // Destrutor para liberar mem ria
20     ~Matriz() {
21         while (head != nullptr) {
22             Node* temp = head;
23             head = head->next;
24             delete temp;
25         }
26     }
27
28     // Inserir m ltiplos elementos no topo da matriz
29     void insereTopo(std::initializer_list<int> values) {
30         for (int value : values) {
31             Node* newNode = new Node(value);
32             newNode->next = head;
33             head = newNode;
34             if (ultimo == nullptr) {
35                 ultimo = head;
36             }
37         }
38     }
39
40     // Inserir m ltiplos elementos no final da matriz
41     void insereFim(std::initializer_list<int> values) {
42         for (int value : values) {
43             Node* newNode = new Node(value);
44             if (ultimo == nullptr) {
45                 head = ultimo = newNode;
46             } else {
47                 ultimo->next = newNode;
48                 ultimo = newNode;
49             }
50         }
51     }
52
53     // Remover m ltiplos elementos do topo da matriz
54     void removeTopo(int count) {
55         while (count-- > 0 && head != nullptr) {
56             Node* temp = head;
57             head = head->next;
58             delete temp;
59             if (head == nullptr) {
60                 ultimo = nullptr;
61             }
62         }
63     }
64
65     // Remover m ltiplos elementos do final da matriz
66     void removeFim(int count) {
67         while (count-- > 0 && head != nullptr) {
68             if (head->next == nullptr) {
69                 delete head;
```

```

70         head = ultimo = nullptr;
71     } else {
72         Node* current = head;
73         while (current->next != ultimo) {
74             current = current->next;
75         }
76         delete ultimo;
77         ultimo = current;
78         ultimo->next = nullptr;
79     }
80 }
81 }
82
83 // M todo para verificar se a matriz cont m um determinado valor
84 bool contem(int value) const {
85     Node* current = head;
86     while (current != nullptr) {
87         if (current->data == value) {
88             return true;
89         }
90         current = current->next;
91     }
92     return false;
93 }
94
95 // Imprimir os elementos da matriz
96 void printMatriz() const {
97     Node* current = head;
98     while (current != nullptr) {
99         std::cout << current->data << " ";
100         current = current->next;
101     }
102     std::cout << std::endl;
103 }
104 };
105
106 int main() {
107     Matriz matriz;
108
109     matriz.inserereTopo({10, 20, 30});
110     matriz.inserereFim({40, 50, 60});
111
112     std::cout << "Matriz: ";
113     matriz.printMatriz();
114
115     matriz.removeTopo(2);
116     std::cout << "Matriz ap s remover K elementos do topo: ";
117     matriz.printMatriz();
118
119     matriz.removeFim(2);
120     std::cout << "Matriz ap s remover K elementos do fim: ";
121     matriz.printMatriz();
122
123     // Verificar se a matriz cont m um valor
124     int valorProcurado = 40;
125     if (matriz.contem(valorProcurado)) {
126         std::cout << "A matriz cont m o valor " << valorProcurado << "." << std::endl;
127     } else {
128         std::cout << "A matriz n o cont m o valor " << valorProcurado << "." << std::endl;
129     }
130
131     return 0;
132 }

```

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 Código

```
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