Documentação de Estruturas de Dados em C++

Alexandre Niess Caio Gomes André Mendes

Rafael Maluf Araujo

10/08/2024

1 Introdução

Este documento apresenta a implementação de quatro estruturas de dados em C++: lista encadeada, pilha, fila e matriz de inteiros. Cada estrutura foi implementada em um arquivo separado e um arquivo principal permite a escolha de operações a serem realizadas nessas estruturas. As sessões abaixo contém detalhes da implementação de cada uma das quatro estruturas, seguido de uma sessão que contém os exemplos de entrada

2 Lista Encadeada

A lista encadeada é uma estrutura de dados linear onde cada elemento aponta para o próximo, formando uma sequência.

2.1 Estrutura do Código

Listing 1: Implementação da Lista Encadeada

```
#include "ListaEncadeada.h"

// declara o no cabeca como nullptr
ListaEncadeada::ListaEncadeada() : cabeca(nullptr) {}

// metodo para inserir valor na lista
void ListaEncadeada::adicionar(int valor) {
    No* novoNo = new No{valor, cabeca};
    cabeca = novoNo;
}

// metodo para busca de valores na lista
bool ListaEncadeada::encontrar(int valor) {
```

```
    // declara o no cabeca como o atual

    No* atual = cabeca;

    // percorre a lista buscando o valor desejado

    while (atual != nullptr) {
        if (atual->dado == valor) {
            return true;
        }
        atual = atual->proximo;
    }
    return false;
}

// metodo para remocao de valores na lista

void ListaEncadeada::remover(int valor) {
    // declara o no cabeca como o atual

    No* atual = cabeca;

    //declara o no anterior como nullptr

    No* anterior = nullptr;

    //percorre lista deslocando os ponteiros para remover logicamente o valor
    desejado

    while (atual != nullptr && atual->dado != valor) {
        anterior = atual;
        atual = atual->proximo;
    }
    if (atual != nullptr) {
        if (anterior == nullptr) {
            cabeca = atual->proximo;
        } else {
            anterior->proximo = atual->proximo;
        }
        delete atual;
    }
}

void ListaEncadeada::removerPrimeiroK(int k) {
    for (int i = 0; i < k && cabeca != nullptr; ++i) {
```

```
        No* temp = cabeca;
        cabeca = cabeca->proximo;
        delete temp;
    }
}
```

3 Pilha

A pilha é uma estrutura de dados linear que segue o princípio LIFO (Last In, First Out).

3.1 Estrutura do Código

Listing 2: Implementação da Pilha

```
#include "Pilha.h"
using namespace std;

// Metodo para inserir valores na pilha
void Pilha::empilhar(int valor) {
    pilha.push(valor);  // Insere o valor no topo da pilha
}

// Metodo para pesquisar o valor na pilha
bool Pilha::encontrar(int valor) {
    stack<int> pilhaTemporaria = pilha;  // Cria uma pilha temporaria
    while (!pilhaTemporaria.empty()) {
        if (pilhaTemporaria.top() == valor) {  // Verifica se o valor
                                               // no topo da pilha
                                               // e o valor procurado
            return true;  // Se encontrar, retorna verdadeiro
        }
        pilhaTemporaria.pop();  // Remove o topo da pilha tempor ria
    }
    return false;  // Se nao encontrar o valor, retorna falso
}

// M todo para remover valor da pilha
void Pilha::desempilhar() {
    if (!pilha.empty()) {  // Verifica se a pilha nao esta vazia
        pilha.pop();  // Remove o valor do topo da pilha
    }
}
```

```
// Metodo para remover os primeiros K elementos da pilha
void Pilha::removerK(int k) {
    // Remove ate K elementos ou ate a pilha ficar vazia
    for (int i = 0; i < k && !pilha.empty(); ++i) {
        pilha.pop();  // Remove o valor do topo da pilha
    }
}
```

4 Fila

A fila é uma estrutura de dados linear que segue o princípio FIFO (First In, First Out).

4.1 Estrutura do Código

Listing 3: Implementação da Fila

```
#include "Fila.h"  // Inclui o cabe alho da classe Fila

// Metodo para inserir valores na fila
void Fila::enfileirar(int valor) {
    fila.push(valor);  // Insere o valor no final da fila
}

// Metodo para pesquisar o valor na fila
bool Fila::encontrar(int valor) {
    queue<int> filaTemp = fila;  // Cria uma fila temporaria
    while (!filaTemp.empty()) {
        // Verifica se o valor no inicio da fila e o valor procurado
        if (filaTemp.front() == valor) {
            return true;  // Se encontrar, retorna verdadeiro
        }
        filaTemp.pop();  // Remove o valor do inicio da fila temporaria
    }
    return false;  // Se nao encontrar o valor, retorna falso
}

// Metodo para remover o valor no inicio da fila
void Fila::desenfileirar() {
    if (!fila.empty()) {  // Verifica se a fila nao esta vazia
        fila.pop();  // Remove o valor do inicio da fila
    }
}

// Implementa  o do metodo que remove os primeiros K elementos da fila
```

```
void Fila::removerK(int k) {
    // Remove ate K elementos ou ate a fila ficar vazia
    for (int i = 0; i < k && !fila.empty(); ++i) {
        fila.pop();  // Remove o valor do inicio da fila
    }
}
```

5 Matriz de Inteiros

A matriz de inteiros é uma estrutura bidimensional que armazena valores inteiros em linhas e colunas.

5.1 Estrutura do Código

Listing 4: Implementação da Matriz de Inteiros

```
#include "Matrix.h"
using namespace std;

// Construtor da classe Matrix que inicializa as
//dimensoes e preenche a matriz com zeros
Matrix::Matrix(int rows, int cols) : rows(rows), cols(cols) {
    // Redimensiona a matriz para as dimensoes
    // especificadas e preenche com zeros
    matrix.resize(rows, vector<int>(cols, 0));
}

// Metodo para setar um valor em uma posicao especifica da matriz
void Matrix::set(int row, int col, int valor) {
    // Verifica se a posicao especificada esta dentro dos limites da matriz
    if (row >= 0 && row < rows && col >= 0 && col < cols) {
        matrix[row][col] = valor; // Define o valor na posicao especificada
    }
}

// Metodo para encontrar um valor na matriz
bool Matrix::encontrar(int valor) {
    // Itera sobre cada elemento da matriz
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            if (matrix[i][j] == valor) {
                return true; // Retorna true se o valor for encontrado
            }
        }
    }
```

```
    return false; // Retorna false se o valor nao for encontrado
}

// Metodo para remover k elementos da matriz
void Matrix::removeK(int k) {
    int removed = 0; // Contador para o numero de elementos removidos
    // Itera sobre a matriz ate que k elementos sejam removidos
    for (int i = 0; i < rows && removed < k; ++i) {
        for (int j = 0; j < cols && removed < k; ++j) {
            if (matrix[i][j] != 0) {
            // Define o elemento como 0, efetivamente removendo-o
                matrix[i][j] = 0;

                // Incrementa o contador de elementos removidos
                ++removed;
            }
        }
    }
}
```

6 Arquivo Principal

O arquivo principal permite a escolha das operações a serem realizadas em cada estrutura.

6.1 Estrutura do Código

Listing 5: Arquivo Principal

```
#include <iostream>
#include "ListaEncadeada.h"
#include "Pilha.h"
#include "Fila.h"
#include "Matrix.h"
using namespace std;

// Funcao que realiza operacoes nas estruturas requisitadas
void performOperations() {
    // Instanciando as estruturas de dados:
    // lista encadeada, pilha, fila e matriz
    ListaEncadeada lista;
    Pilha pilha;
    Fila fila;
    Matrix matrix(3, 3);
```

```
int choice, n, k, valor, row, col; // Variaveis auxiliares

while (true) { // Loop infinito ate o usuario escolher sair

    cout << "Escolha uma opcao:\n";
    cout << "1. Adicionar elementos\n";
    cout << "2. Encontrar um elemento\n";
    cout << "3. Remover elementos\n";
    cout << "4. Sair\n";
    cin >> choice;


    if (choice == 4) break;

switch (choice) {
    case 1:

        // Adicionar elementos as estruturas
        cout << "Digite a quantidade de numeros que quer adicionar: ";
        cin >> n;
        for (int i = 0; i < n; ++i) {
            cout << "Digitar valor: ";
            cin >> valor;
            lista.adicionar(valor); // Adiciona o valor
                                    // na lista encadeada

            pilha.empilhar(valor);  // Empilha o valor na pilha

            fila.enfileirar(valor); // Enfileira o valor na fila

            cout << "Digite a linha e a coluna da matriz: ";

            cin >> row >> col;

            matrix.set(row, col, valor); // Define o valor na posicao
                                         // especificada da matriz
            }
        break;
    case 2:
        // Procurar um elemento nas estruturas
        cout << "Digite valor para pesquisar: ";
        cin >> valor;
        cout << "Na lista: " << (lista.encontrar(valor) ? "Sim" : "Nao")
        << "\n";
        // Checa se o valor esta na lista
        cout << "Na pilha: " << (pilha.encontrar(valor) ? "Sim" : "Nao")
```

```
            << "\n";
            // Checa se o valor esta na pilha
            cout << "Na fila: " << (fila.encontrar(valor) ? "Sim" : "Nao") <<
            "\n";
            // Checa se o valor esta na fila
            cout << "Na matriz: " << (matrix.encontrar(valor) ? "Sim" : "Nao")
            << "\n";
            // Checa se o valor esta na matriz
            break;
        case 3:
            // Remover elementos das estruturas
            cout << "Digite a quantidade de elementos que deseja remover: ";
            cin >> n;
            for (int i = 0; i < n; ++i) {
                cout << "Digite o valor " << i + 1 << " a ser removido: ";
                cin >> k;
                lista.removerPrimeiroK(k);
                pilha.removerK(k);
                fila.removerK(k);
                matrix.removeK(k);
            }
            break;
        default:
            // Caso o usuario escolha uma opcao invalida
            cout << "Escolha invalida\n";
    }
}
}

int main() {
    performOperations(); // Chama a funcao que realiza as operacoes
    return 0;
}
```

Exemplo de Entrada

Aqui está um exemplo de entrada para o nosso programa em C++:

```
// Exemplo de Entrada:
3 //quantidade de elementos que será inserida
10 //primeiro valor
0 1 //posição da matriz para inserir o elemento
20 //segundo valor
1 0 //posição da matriz para inserir o elemento
20 //terceiro valor
```

```
1 0 //posição da matriz para inserir o elemento
```

Este exemplo mostra como o programa deve receber os dados de entrada. A primeira linha contém um número inteiro que indica a quantidade de elementos, seguido pelos próprios elementos. Depois, temos um número inteiro que indica a quantidade de pares de índices e, finalmente, os pares de índices.