LISTAS

Simplesmente e Duplamente Encadeadas

Introdução

As listas encadeadas e duplamente encadeadas são estruturas de dados comuns utilizadas em programação para armazenar e organizar uma coleção de elementos de forma sequencial. Nesta apostila, vamos explorar como implementar essas estruturas em JavaScript.

Listas Encadeadas

2.1. Conceito

Uma lista encadeada consiste em uma sequência de nós, onde cada nó contém um elemento e uma referência para o próximo nó na lista. O último nó geralmente aponta para null, indicando o final da lista.

2.2. Implementação

Para implementar uma lista encadeada em JavaScript, podemos criar uma classe chamada LinkedList, contendo os seguintes métodos:

Node(element): Classe para representar um nó na lista encadeada.
LinkedList(): Construtor da lista encadeada.
append(element): Adiciona um elemento no final da lista.
insert(position, element): Insere um elemento em uma posição específica da lista.
remove(element): Remove um elemento da lista.
indexOf(element): Retorna a posição de um elemento na lista.
isEmpty(): Verifica se a lista está vazia.
size(): Retorna o tamanho da lista.
toString(): Retorna uma representação em string da lista.

Aqui está uma implementação básica de uma lista encadeada em JavaScript:

```
class Node {
  constructor(element) {
    this.element = element;
    this.next = null;
  }
}

class LinkedList {
  constructor() {
    this.head = null;
    this.length = 0;
  }

  append(element) {
    const newNode = new Node(element);

    if (this.head === null) {
      this.head = newNode;
    } else {
      let current = this.head;

      while (current.next !== null) {
        current = current.next;
      }

      current.next = newNode;
    }

    this.length++;
  }


```

```
insert(position, element) {
    if (position >= 0 && position <= this.length) {
      const newNode = new Node(element);

      if (position === 0) {
        newNode.next = this.head;
        this.head = newNode;
      } else {
        let current = this.head;
        let previous = null;
        let index = 0;

        while (index < position) {
          previous = current;
          current = current.next;
          index++;
        }

        newNode.next = current;
        previous.next = newNode;
      }

      this.length++;
      return true;
    }

    return false;
  }
```

```
remove(element) {
    let current = this.head;
    let previous = null;

    while (current !== null) {
      if (current.element === element) {
        if (previous === null) {
          this.head = current.next;
        } else {
          previous.next = current.next;
        }

        this.length--;
        return true;
      }

      previous = current;
      current = current.next;
    }

    return false;
  }

  indexOf(element) {
    let current = this.head;
    let index = 0;

    while (current !== null) {
      if (current.element === element) {
        return index;
      }

      index++;
      current = current.next;
    }

    return -1;
  }
```

```
isEmpty() {
    return this.length === 0;
  }

  size() {
    return this.length;
  }

  toString() {
    let current = this.head;
    let string = '';

    while (current !== null) {
      string += current.element + ' ';
      current = current.next;
    }

    return string.trim();
  }
}
```

Conclusão

Nesta apostila, você aprendeu sobre as listas simplesmente encadeadas em JavaScript. Essa estrutura de dados é fundamental para armazenar e organizar elementos em uma sequência encadeada. Você viu como implementar as operações básicas, como adicionar, inserir, remover, buscar e obter o tamanho da lista.
As listas simplesmente encadeadas são flexíveis e eficientes para manipular elementos em uma ordem específica. Elas são especialmente úteis quando a ordem de inserção e remoção de elementos é frequente. No entanto, elas podem ser menos eficientes para acessar elementos em posições aleatórias, uma vez que requerem percorrer a lista a partir do início.

Lembre-se de que essa implementação básica pode ser aprimorada e adaptada para atender a requisitos específicos ou cenários mais complexos. A compreensão das listas simplesmente encadeadas é fundamental para se tornar um programador eficiente e resolver problemas de forma eficaz.

Continue praticando e explorando diferentes implementações e aplicações das listas simplesmente encadeadas para aprimorar suas habilidades em JavaScript.