Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Aragón

Ingeniería en Computación

Asignatura: Estructura de datos

TAREA 5: Implementación de la Lista doblemente Ligada

Profesor: Jesús Hernández Cabrera

Alumno: Juan Diego Ortiz Cruz

Grupo: 1360

Fecha: 06/09/2024

Capturas Clase NodoDoble:

```typescript
export class NodoDoble<T> {
    private dato: T;
    private siguiente: NodoDoble<T> | null;
    private anterior: NodoDoble<T> | null;

    constructor();
    constructor(dato: T);
    constructor(dato: T, siguiente: NodoDoble<T>, anterior: NodoDoble<T>);
    constructor(dato?: T, siguiente?: NodoDoble<T>, anterior?: NodoDoble<T>) {
        this.dato = dato as T;
        this.siguiente = siguiente || null;
        this.anterior = anterior || null;
    }

    /**
     * getDato
    */
    public getDato(): T {
        return this.dato;
    }

    /**
     * setDato
     */
    public setDato(dato: T): void {
        this.dato = dato;
    }

    /**
     * getSiguiente
     */
    public getSiguiente(): NodoDoble<T> | null {
        return this.siguiente;
    }

    /**
     * setSiguiente
     */
    public setSiguiente(siguiente: NodoDoble<T> | null): void {
        this.siguiente = siguiente;
    }
```

```
1
2       /**
3        * getAnterior
4        */
5       public getAnterior(): NodoDoble<T> | null {
6           return this.anterior;
7       }
8
9       /**
10       * setAnterior
11       */
12      public setAnterior(anterior: NodoDoble<T> | null): void {
13          this.anterior = anterior;
14      }
15
16      public toString(): string {
17          return `<--| ${this.dato} |-->`;
18      }
19  }
```

Capturas Clase DoubleLinkedList:

```
1   import { NodoDoble } from "./NodoDoble";
2
3   export class DoubleLinkedList<T> {
4       private head: NodoDoble<T> | null;
5       private tail: NodoDoble<T> | null;
6       private length: number;
7
8       constructor() {
9           this.head = null;
10          this.tail = null;
11          this.length = 0;
12      }
13
```

```
1
2  /**
3   * estaVacia
4   */
5  public estaVacia(): boolean {
6      let resultado = false;
7      if (this.head === null && this.tail === null) {
8          resultado = true;
9      }
10     return resultado;
11 }
12
13 /**
14  * getLongitud
15  */
16 public getLongitud(): number {
17     return this.length;
18 }
19
20 /**
21  * agregarAlInicio
22  */
23 public agregarAlInicio(valor: T): void {
24     const nuevoNodo = new NodoDoble<T>(valor);
25     if (this.estaVacia()) {
26         this.head = nuevoNodo;
27         this.tail = nuevoNodo;
28     } else {
29         this.head!.setAnterior(nuevoNodo);
30         nuevoNodo.setSiguiente(this.head!);
31         this.head = nuevoNodo;
32     }
33     this.length++;
34 }
35
36 /**
37  * agregarAlFinal
38  */
39 public agregarAlFinal(valor: T): void {
40     const nuevoNodo = new NodoDoble<T>(valor);
41     if (this.estaVacia()) {
42         this.head = nuevoNodo;
43         this.tail = nuevoNodo;
44     } else {
45         this.tail!.setSiguiente(nuevoNodo);
46         nuevoNodo.setAnterior(this.tail!);
47         this.tail = nuevoNodo;
48     }
49     this.length++;
50 }
51
```

```
1
2  /**
3   * agregarDespuesDe
4   */
5  public agregarDespuesDe(referencia: T, valor: T): void {
6      const nuevoNodo = new NodoDoble<T>(valor);
7      let aux = this.head;
8      while (aux!.getDato() !== referencia) {
9          aux = aux!.getSiguiente();
10     }
11     if (aux === null) {
12         console.log("Referencia no encontrada");
13     } else {
14         nuevoNodo.setSiguiente(aux.getSiguiente()!);
15         nuevoNodo.setAnterior(aux);
16         aux.getSiguiente()!.setAnterior(nuevoNodo);
17         aux.setSiguiente(nuevoNodo);
18         this.length++;
19     }
20 }
21
22 /**
23  * obtener
24  */
25 public obtener(posicion: number): T | null {
26     if (posicion < 0 || posicion >= this.length) {
27         return null;
28     }
29     let aux = this.head;
30     for (let i = 0; i < posicion; i++) {
31         aux = aux!.getSiguiente();
32     }
33     return aux!.getDato();
34 }
35
36 /**
37  * eliminarElPrimero
38  */
39 public eliminarElPrimero(): void {
40     if (this.estaVacia()) {
41         return;
42     }
43     if (this.head === this.tail) {
44         this.head = null;
45         this.tail = null;
46     } else {
47         this.head = this.head!.getSiguiente();
48         this.head!.setAnterior(null);
49     }
50     this.length--;
51 }
52
```

```
1
2  /**
3   * eliminarElUltimo
4   */
5  public eliminarElUltimo(): void {
6      if (this.estaVacia()) {
7          return;
8      }
9      if (this.head === this.tail) {
10         this.head = null;
11         this.tail = null;
12     } else {
13         this.tail = this.tail!.getAnterior();
14         this.tail!.setSiguiente(null);
15     }
16     this.length--;
17 }
18
19 /**
20  * eliminar
21  */
22 public eliminar(posicion: number): void {
23     if (posicion < 0 || posicion >= this.length) {
24         return;
25     }
26     if (posicion === 0) {
27         this.eliminarElPrimero();
28         return;
29     }
30     if (posicion === this.length - 1) {
31         this.eliminarElUltimo();
32         return;
33     }
34     let aux = this.head;
35     for (let i = 0; i < posicion; i++) {
36         aux = aux!.getSiguiente();
37     }
38     aux!.getAnterior()!.setSiguiente(aux!.getSiguiente()!);
39     aux!.getSiguiente()!.setAnterior(aux!.getAnterior()!);
40     this.length--;
41 }
42
43 /**
44  * actualizar
45  */
46 public actualizar(aBuscar: T, valor: T): void {
47     let aux = this.head;
48     while (aux !== null) {
49         if (aux.getDato() === aBuscar) {
50             aux.setDato(valor);
51             return;
52         }
53         aux = aux.getSiguiente();
54     }
55 }
56
```

```
1
2       /**
3        * buscar
4        */
5       public buscar(valor: T): number {
6           let aux = this.head;
7           let posicion = 0;
8           while (aux !== null) {
9               if (aux.getDato() === valor) {
10                  return posicion;
11              }
12              aux = aux.getSiguiente();
13              posicion++;
14          }
15          return -1;
16      }
17
18      /**
19       * transversal @param direccion 0 --> izq a derecha  si es 1 --> derecha a izq
20       */
21      public transversal(direccion: number = 0): void {
22          if (direccion === 1) {
23              let aux = this.head;
24              process.stdout.write("|");
25              while (aux !== null) {
26                  process.stdout.write(`${aux.getDato()}| <-> |`);
27                  aux = aux.getAnterior();
28              }
29              console.log("null|");
30          } else {
31              let aux = this.head;
32              process.stdout.write("|");
33              while (aux !== null) {
34                  process.stdout.write(`${aux.getDato()}| <-> |`);
35                  aux = aux.getSiguiente();
36              }
37              console.log("null|");
38          }
39      }
40  }
```

Capturas Main:

```ts
import { DoubleLinkedList } from "./DoubleLinkedList";

function main() {
    const lista = new DoubleLinkedList<number>();

    // Agregar al inicio el 50
    lista.agregarAlInicio(50);

    // Agregar al final el 60, 65, 70, 80 y 90
    lista.agregarAlFinal(60);
    lista.agregarAlFinal(65);
    lista.agregarAlFinal(70);
    lista.agregarAlFinal(80);
    lista.agregarAlFinal(90);

    // Imprimir el contenido
    console.log("Contenido inicial:");
    lista.transversal();

    // Eliminar el de la posición 2
    lista.eliminar(2);

    // Volver a imprimir el contenido
    console.log("\nContenido después de eliminar la posición 2:");
    lista.transversal();

    // Actualizar el cuarto elemento a 88
    lista.actualizar(80, 88);

    // Volver a imprimir el contenido
    console.log("\nContenido después de actualizar el cuarto elemento a 88:");
    lista.transversal();

    // Buscar el valor 80 e imprimir en qué posición se encuentra
    let posicion = lista.buscar(80);
    console.log(`\nEl valor 80 se encuentra en la posición: ${posicion}`);

    // Buscar el valor 88 e imprimir en qué posición se encuentra
    posicion = lista.buscar(88);
    console.log(`\nEl valor 88 se encuentra en la posición: ${posicion}`);
}

main();
```

Capturas consola: