**Курс " Javascript Fullstack Developer" и "Javascript Frontend"**

# **Кожевников Сергей 2024 год**

# **Задание 4**

**Сделать как минимум 2 пункта (при желании можно сделать все пункты):**  
  
1) Написать класс, реализующий двусвязный список. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения длины списка.  
2) Написать класс, реализующий взвешенный граф. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения количества элементов. Предусмотреть метод поиска кратчайшего расстояния между двумя узлами.  
3) Написать класс, реализующий AVL дерево. Предусмотреть методы поиска, вставки, удаления, изменения элемента.  
4) Написать класс, реализующий красно-чёрное дерево. Предусмотреть методы поиска, вставки, удаления, изменения элемента.  
5) Написать класс, реализующий дерево. Предусмотреть методы поиска, вставки элемента и определения высоты дерева.  
6) Написать класс, реализующий бинарное дерево. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения высоты дерева.  
7) Написать класс, реализующий список. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения длины списка.  
8) Написать класс, реализующий граф. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения количества элементов. Предусмотреть метод поиска кратчайшего расстояния между двумя узлами.  
9) Реализовать класс, реализующие хеш-таблицу. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения количества элементов.  
10) Программа получает на вход список 10 городов. Далее вводятся строки типа “<город1>;<город2>=<цена проезда>”, определяющие цену дороги от одного города к другому. Последняя строка - “<городA>;<городB>”. Программа должна определить наименьшую стоимость проезда от городаА до городаB. (если таких дорог не существует, то выведите стоимость “undefined”)

**Опциональное дз:**

Установить NodeJS, если он еще у вас не установлен

Посмотреть в интернете, как запускать скрипты в NodeJS

Ознакомиться с вводом-выводом браузера (alert, prompt, confirm)

Посмотреть в интернете, как можно реализовать ввод-вывод в NodeJS

Написать программу, подсчитывающую количество простых чисел от a до b (даются на

вход). Реализовать ввод/вывод отдельно под браузерную и NodeJS среды.

Опциональное дз:

Изучить работу прототипов.

Обратить внимание, что при создании ES6 класса методы записываются в прототип

объекта, а не в сам объект.

# **Решение**

Установи NodeJS.

Посмотрел в интернете, как запускать скрипты в NodeJS

**PS D:\JSProject> node app.js**

Ознакомился с вводом-выводом браузера (alert, prompt, confirm).

Посмотрел в интернете, как можно реализовать ввод-вывод в NodeJS.

Написал программу, подсчитывающую количество простых чисел от a до b (даются на

вход). Реализовал ввод/вывод отдельно под браузерную и NodeJS среды.

Для браузера

var \_a = prompt('Введите число', 10);

console.log('a = ' + \_a);

let \_start = 2,

\_end = \_a;

for (let i = \_start; i <= \_end; i++)

{

    let edge = Math.sqrt(i);

    let isPrime = true;

    for (let j = 2; j <= edge; j++) {

        if (i % j) continue;

        isPrime = false;

        break;

    }

    if (isPrime)

    //Session.Output("Простое число " + "[" + i +"] = " + + i);

    {

        alert(i);

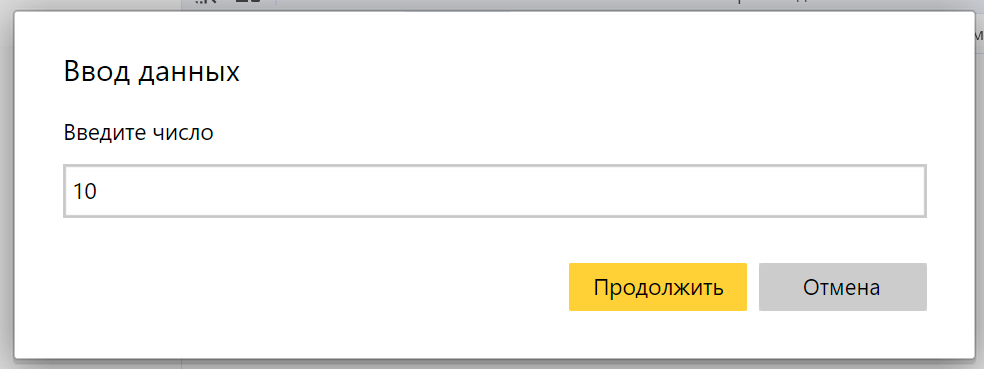
        console.log('i = ' + i);

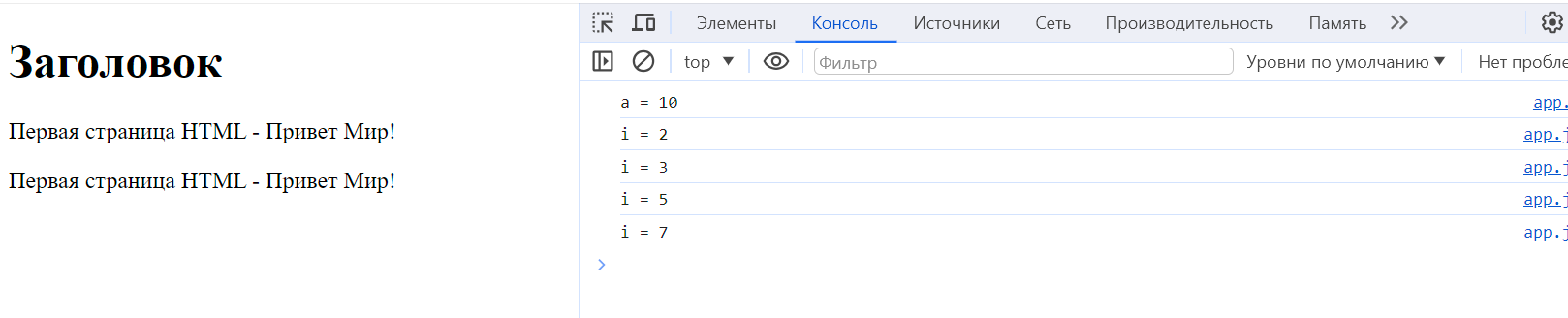
        var \_b = confirm("Результат правильный ?");

        //alert( \_b );

    }

}





Для Nodejs

     let \_start = 2,

     \_end = 20;

for (let i = \_start; i <= \_end; i++)

{

    let edge = Math.sqrt(i);

    let isPrime = true;

    for (let j = 2; j <= edge; j++) {

        if (i % j) continue;

        isPrime = false;

        break;

    }

    if (isPrime)

    //Session.Output("Простое число " + "[" + i +"] = " + + i);

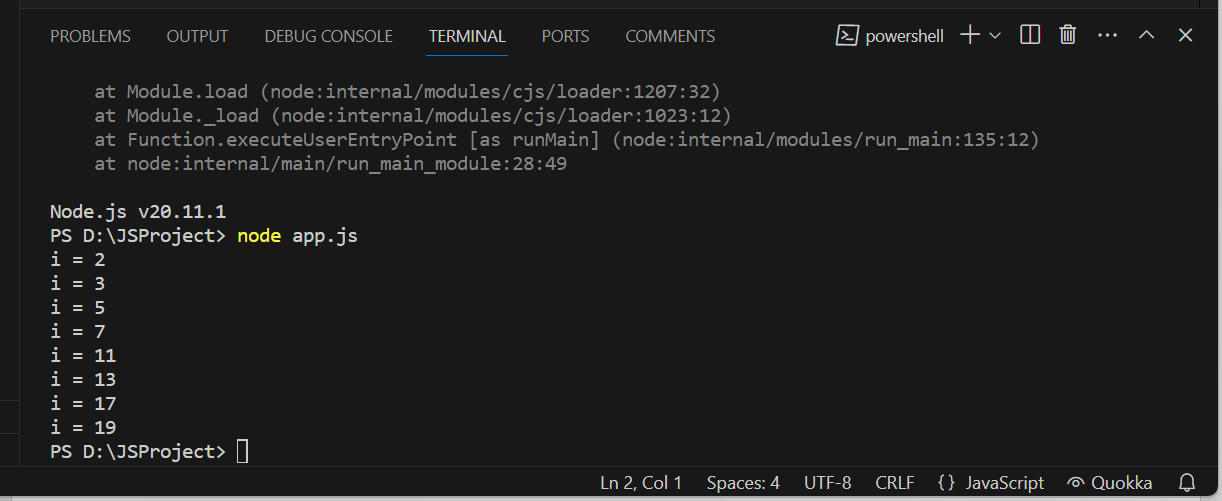
    {

        //alert(i);

        console.log('i = ' + i);

    }

}



## Написать класс, реализующий двусвязный список. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения длины списка.

## **Реализация двусвязного списка**

// Doubly Linked list Node

class Node {

  // Constructor to create a new node

  // next and prev is by default initialized as null

  constructor(val) {

      // To store the value

      this.data = val;

      // To link the next Node

      this.next = null;

      // TO link the previous Node

      this.prev = null;

  }

}

// Doubly Linked List

class DoublyLinkedList {

  // Constructor to create a new linked list

  constructor() {

      // To contain the first item of the list

      this.head = null;

      // To contain the last item of the list

      this.tail = null;

  }

  // To check if the list is empty

  isEmpty() {

      if (this.head == null) return true;

      return false;

  }

  // Method to add item at the last of doubly linked list

  addItem(val) {

      // Create a temporary variable

      let temp = new Node(val);

      // If the list is empty link assign

      // new node to both head and tail

      if (this.head == null) {

          this.head = temp;

          this.tail = temp;

      }

      // else add item to the tail and shift tail

      else {

          this.tail.next = temp;

          this.tail = this.tail.next;

      }

  }

  // To traverse and display the list

  display() {

      // Check if the List is empty

      if (!this.isEmpty()) {

          // traverse the list using new current pointer

          let curr = this.head;

          while (curr !== null) {

              // Display element

              console.log(curr.data);

              // Shift the current pointer

              curr = curr.next;

          }

      }

  }

}

// Create new Doubly Linked List

const dll = new DoublyLinkedList();

// Add elements in the list

dll.addItem(15);

dll.addItem(17);

dll.addItem(19);

dll.addItem(21);

dll.addItem(25);

dll.addItem(27);

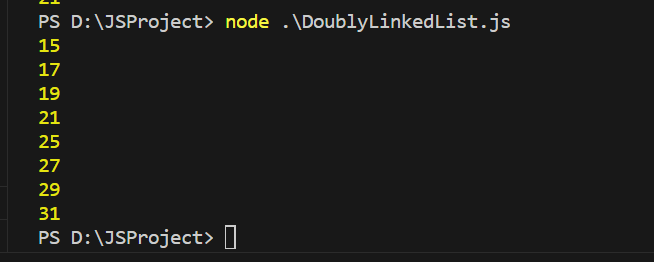
dll.addItem(29);

dll.addItem(31);

// Display the list

dll.display();

Результат работы



7) Написать класс, реализующий список. Предусмотреть методы поиска, вставки, удаления, изменения элемента и определения длины списка.

class Node {

    // constructor

    constructor(element) {

        this.element = element;

        this.next = null

    }

}

// linkedlist class

class LinkedList {

    constructor() {

        this.head = null;

        this.size = 0;

    }

    // adds an element at the end

    // of list

    add(element) {

        // creates a new node

        let node = new Node(element);

        // to store current node

        let current;

        // if list is Empty add the

        // element and make it head

        if (this.head == null)

            this.head = node;

        else {

            current = this.head;

            // iterate to the end of the

            // list

            while (current.next) {

                current = current.next;

            }

            // add node

            current.next = node;

        }

        this.size++;

    }

    // insert element at the position index

    // of the list

    insertAt(element, index) {

        if (index < 0 || index > this.size)

            return console.log("Please enter a valid index.");

        else {

            // creates a new node

            let node = new Node(element);

            let curr, prev;

            curr = this.head;

            // add the element to the

            // first index

            if (index == 0) {

                node.next = this.head;

                this.head = node;

            } else {

                curr = this.head;

                let it = 0;

                // iterate over the list to find

                // the position to insert

                while (it < index) {

                    it++;

                    prev = curr;

                    curr = curr.next;

                }

                // adding an element

                node.next = curr;

                prev.next = node;

            }

            this.size++;

        }

    }

    // removes an element from the

    // specified location

    removeFrom(index) {

        if (index < 0 || index >= this.size)

            return console.log("Please Enter a valid index");

        else {

            let curr, prev, it = 0;

            curr = this.head;

            prev = curr;

            // deleting first element

            if (index === 0) {

                this.head = curr.next;

            } else {

                // iterate over the list to the

                // position to remove an element

                while (it < index) {

                    it++;

                    prev = curr;

                    curr = curr.next;

                }

                // remove the element

                prev.next = curr.next;

            }

            this.size--;

            // return the remove element

            return curr.element;

        }

    }

    // removes a given element from the

    // list

    removeElement(element) {

        let current = this.head;

        let prev = null;

        // iterate over the list

        while (current != null) {

            // comparing element with current

            // element if found then remove the

            // and return true

            if (current.element === element) {

                if (prev == null) {

                    this.head = current.next;

                } else {

                    prev.next = current.next;

                }

                this.size--;

                return current.element;

            }

            prev = current;

            current = current.next;

        }

        return -1;

    }

    // finds the index of element

    indexOf(element) {

        let count = 0;

        let current = this.head;

        // iterate over the list

        while (current != null) {

            // compare each element of the list

            // with given element

            if (current.element === element)

                return count;

            count++;

            current = current.next;

        }

        // not found

        return -1;

    }

    // checks the list for empty

    isEmpty() {

        return this.size == 0;

    }

    // gives the size of the list

    size\_of\_list() {

        console.log(this.size);

    }

    // prints the list items

    printList() {

        let curr = this.head;

        let str = "";

        while (curr) {

            str += curr.element + " ";

            curr = curr.next;

        }

        console.log(str);

    }

}

// creating an object for the

// Linkedlist class

let ll = new LinkedList();

// testing isEmpty on an empty list

// returns true

console.log(ll.isEmpty());

// adding element to the list

ll.add(10);

// prints 10

ll.printList();

// returns 1

console.log(ll.size\_of\_list());

// adding more elements to the list

ll.add(20);

ll.add(30);

ll.add(40);

ll.add(50);

// returns 10 20 30 40 50

ll.printList();

// prints 50 from the list

console.log("is element removed ?" + ll.removeElement(50));

// prints 10 20 30 40

ll.printList();

// returns 3

console.log("Index of 40 " + ll.indexOf(40));

// insert 60 at second position

// ll contains 10 20 60 30 40

ll.insertAt(60, 2);

ll.printList();

// returns false

console.log("is List Empty ? " + ll.isEmpty());

// remove 3rd element from the list

console.log(ll.removeFrom(3));

// prints 10 20 60 40

ll.printList();

