|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_ Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_ Программное обеспечение ЭВМ и информационные технологии \_\_\_\_\_\_

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

Дисциплина Архитектура ЭВМ

Студент Мишина Елена Викторовна

*фамилия, имя, отчество*

Группа ИУ7-54Б

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Попов А.Ю.

*подпись, дата фамилия, и.о.*

*2020 г.*

Оглавление

[Цель работы: 3](#_Toc51238076)

[Task 1 4](#_Toc51238077)

[1. Задание 1 4](#_Toc51238078)

[2. Задание 2 9](#_Toc51238079)

[3. Задание 3 13](#_Toc51238080)

[Task 2 20](#_Toc51238081)

[1. Задание 1 20](#_Toc51238082)

[2. Задание 2 22](#_Toc51238083)

[3. Задание 3 25](#_Toc51238084)

[Вывод: 29](#_Toc51238085)

# 

# Цель работы:

Приобретение базовых знаний JavaScript, написание программ, демонстрирующих знание циклов, строк, массивов, объектов и функций, получение знаний в области ООП языка JavaScript.

# Task 1

# Задание 1

**Условие задачи:**

Создать хранилище в оперативной памяти для хранения информации о детях.

Необходимо хранить информацию о ребенке: фамилия и возраст.

Необходимо обеспечить уникальность фамилий детей.

Реализовать функции:

* CREATE READ UPDATE DELETE для детей в хранилище
* Получение среднего возраста детей
* Получение информации о самом старшем ребенке
* Получение информации о детях, возраст которых входит в заданный отрезок
* Получение информации о детях, фамилия которых начинается с заданной буквы
* Получение информации о детях, фамилия которых длиннее заданного количества символов
* Получение информации о детях, фамилия которых начинается с гласной буквы

**Листинг программы:**

"use strict";

let children = new Map()

function CREATE(surname, age) {

   return children.set(surname, age)

}

function READ(surname) {

    return children.get(surname)

}

function UPDATE(surname, age) {

    return children.set(surname, age)

}

function DELETE(surname) {

    return children.delete(surname)

}

function average\_age() {

    if (children.size === 0)

        return false

    let average = 0

    for(let age of children.values()){

        average += age

    }

    return average / children.size

}

function max\_age\_child() {

    let max\_age = -1

    let max\_child

    for(let child of children.entries()){

         if (child[1] > max\_age) {

             max\_age = child[1]

             max\_child = child

         }

    }

    return max\_child

}

function period\_age(min\_age, max\_age) {

    let period\_children = []

    for(let child of children.entries()) {

        if (min\_age <= child[1] && child[1] <= max\_age)

            period\_children.push(child)

    }

    return period\_children

}

function right\_letter(letter) {

    let letter\_children = []

    for(let child of children.entries()) {

        if (child[0][0].toLowerCase() === letter.toLowerCase())

            letter\_children.push(child)

    }

    return letter\_children

}

function long\_name(size) {

    let long\_name\_children = []

    for(let child of children.entries()) {

        if (child[0].length > size)

            long\_name\_children.push(child)

    }

    return long\_name\_children

}

function vowel\_letter() {

    let  vowel\_letter\_children= []

    const vowels = ['a', 'e', 'i', 'o', 'u']

    for(let child of children.entries()) {

        if (vowels.includes(child[0][0].toLowerCase()))

            vowel\_letter\_children.push(child)

    }

    return  vowel\_letter\_children

}

**Тестирование:**

PS C:\Users\mishi\Desktop\EVM\prtoject\_text\_1> node index.js

Average age:

8.2

Max age:

[ 'Efbnn', 12 ]

Period age [1, 11]:

[ [ 'sjksl', 3 ], [ 'fd', 11 ], [ 'ffd1', 10 ], [ 'Adfg',

5 ] ]

Surname rigth letter:

[ [ 'fd', 11 ], [ 'ffd1', 10 ] ]

Surname is longer than 5:

[ [ 'sjksl', 3 ], [ 'ffd1', 10 ], [ 'Adfg', 5 ], [ 'Efbnn', 12 ] ]

Surname starts with a vowel:

[ [ 'Adfg', 5 ], [ 'Efbnn', 12 ] ]

# Задание 2

**Условие задачи:**

Создать хранилище в оперативной памяти для хранения информации о студентах.

Необходимо хранить информацию о студенте: название группы, номер студенческого билета, оценки по программированию.

Необходимо обеспечить уникальность номеров студенческих билетов.

Реализовать функции:

* CREATE READ UPDATE DELETE для студентов в хранилище
* Получение средней оценки заданного студента
* Получение информации о студентах в заданной группе
* Получение студента, у которого наибольшее количество оценок в заданной группе
* Получение студента, у которого нет оценок

**Листинг программы:**

"use strict";

class Student {

    constructor(group\_name, student\_card, marks) {

        this.group\_name = group\_name

        this.student\_card = student\_card

        this.marks = marks

    }

    get output(){

        return `${this.group\_name} ${this.student\_card} ${this.marks}`;

    }

}

class Students{

    constructor() {

        this.arr = []

    }

    CREATE(person){

        for (const man of this.arr){

            if (man.student\_card === person.student\_card){

                return false

            }

        }

        this.arr.push(person)

        return true

    }

    READ(person){

        let flag = 1

        for (const man of this.arr){

            if (man.student\_card === person.student\_card){

                flag = 0

            }

        }

        if (flag === 0){

            console.log("NOT UNIQIE STUDENT CARD")

            return false

        }

        console.log(person.output)

        return true

    }

    UPDATE(person){

        let j = -1

        for (let i = 0; i < this.arr.length; i++){

            if (arr[i].student\_card === person.student\_card){

                j = i

            }

        }

        if (j === -1) {

            console.log("NOT UNIQIE STUDENT CARD")

            return false

        }

        this.arr.splice(j, 1, st)

        return true

    }

    DELETE(person){

        let j = -1

        for (let i = 0; i < this.arr.length; i++){

            if (arr[i].group\_name === person.group\_name){

                j = i

            }

        }

        if (j === -1) {

            console.log("ERROR")

            return false

        }

        this.arr.splice(j, 1)

        return true

    }

    average\_mark\_st(person){

        let n = 0

        let i = 0

        for (const mark of person.marks) {

            i++;

            n += mark

        }

        if (i === 0){

            console.log("ERROR")

            return false

        }

        return n / i

    }

    info\_group(group){

        for (const man of this.arr){

            if (man.group\_name === group){

                console.log(man.output)

            }

        }

        return true

    }

    info\_st\_group(group){

        let len = -1

        let best\_person = undefined

        for (const man of this.arr){

            if(man.group\_name === group)

            {

                if (man.marks.length > len){

                    len = man.marks.length

                    best\_person = man

                }

            }

        }

        if (len === -1){

            console.log("ERROR")

            return false

        }

        console.log(best\_person.output)

        return true

    }

    info\_zero\_st(){

        for (const man of this.arr){

            if (man.marks.length === 0){

                console.log(man.output)

            }

        }

        return true

    }

}

**Тестирование:**

function main() {

    student\_1 = new Student("VVVV", 15, [5,3,5])

    student\_2 = new Student("VVVV", 16, [5,3,5,4,5])

    student\_3 = new Student("XXXX", 17, [4,5,6,7,8])

    student\_4 = new Student("XXXX", 19, [5,3,4])

    students\_database = new Students()

    students\_database.CREATE(student\_1)

    students\_database.CREATE(student\_2)

    students\_database.CREATE(student\_3)

    students\_database.CREATE(student\_4)

    console.log(students\_database.average\_mark\_st(student\_1))

    students\_database.info\_st\_group("XXXX")

    students\_database.info\_group("XXXX")

    students\_database.info\_zero\_st()

}

main()

PS C:\Users\mishi\Desktop\EVM\task1\_2> node task1\_2.js

4.333333333333333

XXXX 17 4,5,6,7,8

XXXX 17 4,5,6,7,8

XXXX 19 5,3,4

# Задание 3

**Условие задачи:**

Создать хранилище в оперативной памяти для хранения точек.

Неоходимо хранить информацию о точке: имя точки, позиция X и позиция Y.

Необходимо обеспечить уникальность имен точек.

Реализовать функции:

* CREATE READ UPDATE DELETE для точек в хранилище
* Получение двух точек, между которыми наибольшее расстояние
* Получение точек, находящихся от заданной точки на расстоянии, не превышающем заданную константу
* Получение точек, находящихся выше / ниже / правее / левее заданной оси координат
* Получение точек, входящих внутрь заданной прямоугольной зоны

**Листинг кода:**

"use strict";

class Dot {

    constructor(name, x, y) {

        this.name = name

        this.x = x

        this.y = y

    }

    get output(){

        return `${this.name} ${this.x} ${this.y}`;

    }

}

class Dots{

    constructor() {

        this.arr = []

    }

    CREATE(dt){

        for (const d of this.arr){

            if (dt.name === d.name){

                return false

            }

        }

        this.arr.push(dt)

        return true

    }

    READ(dt){

        let flag = 1

        for (const d of this.arr){

            if (dt.name === d.name){

                flag = 0

            }

        }

        if (flag === 1){

            console.log("Invalid read dot name")

            return false

        }

        dt.output

        return true

    }

    UPDATE(dt){

        let j = -1

        for (let i = 0; i < this.arr.length; i++){

            if (this.arr[i].name === dt.name){

                j = i

            }

        }

        if (j === -1) {

            console.log("Invalid read dot name")

            return false

        }

        this.arr.splice(j, 1, dt)

        return true

    }

    DELETE(dt){

        let j = -1

        for (let i = 0; i < this.arr.length; i++){

            if (this.arr[i].name === dt.name){

                j = i

            }

        }

        if (j === -1) {

            console.log("Invalid read dot name")

            return false

        }

        this.arr.splice(j, 1)

        return true

    }

    max\_range(){

        if (this.arr.length <= 1){

            console.log("Invalid read array")

            return false

        }

        let max = 0

        let temp\_max = 0

        let log = []

        for (let i = 0; i < this.arr.length - 1; i++){

            temp\_max = Math.sqrt(Math.pow(this.arr[i].x - this.arr[i + 1].x, 2) +

                Math.pow(this.arr[i].y - this.arr[i + 1].y, 2))

            if (temp\_max > max){

                max = temp\_max

                log[0] = this.arr[i]

                log[1] = this.arr[i + 1]

            }

        }

        return log

    }

    near\_dots(dt, len){

        if (len <= 0) {

            console.log("Invalid read len")

            return false

        }

        let temp\_len = 0

        let log = []

        for (let i = 0; i < this.arr.length; i++){

            temp\_len = Math.sqrt(Math.pow(dt.x - this.arr[i].x, 2) + Math.pow(dt.y - this.arr[i].y, 2))

            if (temp\_len < len){

                log.push(this.arr[i])

            }

        }

        return log

    }

    get\_dot\_axes(d){

        let log = []

        for (let i = 0; i < this.arr.length; i++) {

            if (this.arr[i].x > d.x && this.arr[i].y > d.y){

                log.push(this.arr[i], "1")

            }

            else if (this.arr[i].x < d.x && this.arr[i].y > d.y){

                log.push(this.arr[i], "2")

            }

            else if (this.arr[i].x < d.x && this.arr[i].y < d.y){

                log.push(this.arr[i], "3")

            }

            else if (this.arr[i].x > d.x && this.arr[i].y < d.y){

                log.push(this.arr[i], "4")

            }

        }

        return log

    }

    inside\_rectangle(dt\_l\_d, dt\_r\_u){

        let log = []

        for (let i = 0; i < this.arr.length; i++) {

            if (this.arr[i].x < dt\_r\_u.x && this.arr[i].x > dt\_l\_d.x &&

                this.arr[i].y < dt\_r\_u.y && this.arr[i].y > dt\_l\_d.y)

                log.push(this.arr[i])

            }

        return log

    }

}

**Тестирование:**

function main(){

    let d1 = new Dot("abc", 123, 1)

    let d11 = new Dot("ab12c", 1323, 3213)

    let d2 = new Dot("qwerty", 523123133, 1)

    let d3 = new Dot("rtyuifgh", 565, 334)

    let d4 = new Dot("rqwe", 55, 23)

    let ds = new Dots()

    ds.CREATE(d1)

    ds.CREATE(d11)

    ds.CREATE(d2)

    ds.CREATE(d3)

    ds.DELETE(d4)

    console.log(ds.READ(d1))

    let d5 = new Dot("qwerty", 5233, 12)

    ds.UPDATE(d5)

    console.log(ds.READ(d2))

    console.log(ds.max\_range())

    console.log(ds.near\_dots(d1, 10000000))

    console.log(ds.get\_dot\_axes(d2))

    console.log(ds.inside\_rectangle(d1, d11))

}

main()

PS C:\Users\mishi\Desktop\EVM\task1\_3> node task1\_3.js

true

true

[

Dot { name: 'ab12c', x: 1323, y: 3213 },

Dot { name: 'qwerty', x: 5233, y: 12 }

]

[

Dot { name: 'abc', x: 123, y: 1 },

Dot { name: 'ab12c', x: 1323, y: 3213 },

Dot { name: 'qwerty', x: 5233, y: 12 },

Dot { name: 'rtyuifgh', x: 565, y: 334 }

]

[

Dot { name: 'ab12c', x: 1323, y: 3213 },

'2',

Dot { name: 'qwerty', x: 5233, y: 12 },

'2',

Dot { name: 'rtyuifgh', x: 565, y: 334 },

'2'

]

[ Dot { name: 'rtyuifgh', x: 565, y: 334 } ]

# Task 2

# Задание 1

**Условие задачи:**

Создать класс Точка.

Добавить классу точка Точка метод инициализации полей и метод вывода полей на экран

Создать класс Отрезок.

У класса Отрезок должны быть поля, являющиеся экземплярами класса Точка.

Добавить классу Отрезок метод инициализации полей, метод вывода информации о полях на экран, а так же метод получения длины отрезка.

**Листинг кода:**

"use strict";

class Dot {

    constructor(x, y) {

        this.x = x;

        this.y = y;

    }

    renderFields() {

        let messageX = "X: " + this.x;

        let messageY = "Y: " + this.y;

        let fullMessage = messageX + " " + messageY;

        console.log(fullMessage);

    }

}

class Line {

    constructor(dot\_1, dot\_2){

        this.dot\_1 = dot\_1

        this.dot\_2 = dot\_2

    }

    renderFields() {

        console.log("DOT №1")

        this.dot\_1.renderFields()

        console.log("DOT №2")

        this.dot\_2.renderFields()

    }

    len\_line() {

        let len = Math.sqrt(Math.pow((this.dot\_2.x-this.dot\_1.x),2) + Math.pow((this.dot\_2.y-this.dot\_1.y),2))

        return len

    }

}

**Тестирование:**

function main(){

    dot\_1 = new Dot(0,0)

    dot\_2 = new Dot(5,0)

    dot\_1.renderFields()

    dot\_2.renderFields()

    line = new Line(dot\_1, dot\_2)

    line.renderFields()

    console.log(line.len\_line())

}

main()

PS C:\Users\mishi\Desktop\EVM\task2\_1> node task.js

X: 0 Y: 0

X: 5 Y: 0

DOT №1

X: 0 Y: 0

DOT №2

X: 5 Y: 0

5

# Задание 2

**Условие задачи:**

Создать класс *Треугольник*.

Класс *Треугольник* должен иметь поля, хранящие длины сторон треугольника.

Реализовать следующие методы:

* Метод инициализации полей
* Метод проверки возможности существования треугольника с такими сторонами
* Метод получения периметра треугольника
* Метод получения площади треугольника
* Метод для проверки факта: является ли треугольник прямоугольным

**Листинг кода:**

"use strict";

class Triangle {

    constructor(a, b, c){

        this.a = a

        this.b = b

        this.c = c

    }

    check\_triangle(){

        if((this.a + this.b) < this.c || (this.a + this.c) < this.b || (this.c + this.b) < this.a){

            console.log("ERROR.NOT TRIANGLE")

            return false

        }

        return true

    }

    perimeter\_triangle(){

        if(this.check\_triangle() === false){

            return false

        }

        return this.a + this.b + this.c

    }

    square\_triangle(){

        if(this.check\_triangle() === false){

            return false

        }

        let p = this.perimeter\_triangle()/2

        let square = Math.sqrt(p\*(p-this.a)\*(p-this.b)\*(p-this.c))

        return square

    }

    rectangular\_triangle(){

        if(this.check\_triangle() === false){

            return false

        }

        let eps = 0.0001

        let ab = Math.pow(this.a, 2) + Math.pow(this.b, 2)

        let ac = Math.pow(this.a, 2) + Math.pow(this.c, 2)

        let bc = Math.pow(this.b, 2) + Math.pow(this.c, 2)

        if(ab - Math.pow(this.c, 2) < eps || ac - Math.pow(this.b, 2) < eps || bc - Math.pow(this.a, 2)){

            console.log("rectangular triangle")

            return true

        }

        console.log("NOT rectangular triangle")

        return false

    }

}

**Тестирование:**

function main(){

    let tr = new Triangle(3, 4, 5)

    let tr\_1 = new Triangle(0, 3, 5)

    tr\_1.check\_triangle()

    tr.check\_triangle()

    console.log(tr.perimeter\_triangle())

    console.log(tr.square\_triangle())

    tr.rectangular\_triangle()

}

main()

node task.js

ERROR.NOT TRIANGLE

12

6

rectangular triangle

# Задание 3

**Условие задачи:**

Реализовать программу, в которой происходят следующие действия:

Происходит вывод целых чисел от 1 до 10 с задержками в 2 секунды.

После этого происходит вывод от 11 до 20 с задержками в 1 секунду.

Потом опять происходит вывод чисел от 1 до 10 с задержками в 2 секунды.

После этого происходит вывод от 11 до 20 с задержками в 1 секунду.

Это должно происходить циклически.

**Листинг кода:**

"use strict";

let i = 0;

let flag = 0;

let interval = setInterval(() => {

    if (!flag) {

        i++;

        console.log(i);

        if (i === 10) {

            flag = 1;

        }

    }

}, 2000);

let interval2 = setInterval(() => {

    if (flag) {

        i++;

        console.log(i);

        if (i === 20) {

            i = 0;

            flag = 0;

        }

    }

}, 1000);

**Тестирование:**

PS C:\Users\mishi\Desktop\EVM\task2\_3> node task.js

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

1

2

3

4

5

# Вывод:

Были приобретены базовые знания JavaScript. Все задачи на знание основ JavaScript, а также ООП в JavaScript выполнены успешно.