|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования 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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ДИСЦИПЛИНА «Архитектура ЭВМ»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 1**

|  |  |
| --- | --- |
| **Студент** \_\_Кобаренков И.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Группа** \_\_ИУ7-51Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Преподаватель** \_\_Попов А. Ю.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |

Москва.

2020 г.

Содержание

[Задание 1.1 3](#_Toc50907040)

[Задание 1.2 7](#_Toc50907041)

[Задание 1.3 11](#_Toc50907042)

[Задание 2.1 15](#_Toc50907043)

[Задание 2.2 16](#_Toc50907044)

[Задание 2.3 17](#_Toc50907045)

Цель работы: изучить основы синтаксиса javascript, познакомиться с фреймворком node.js, изучить стрелочные функции и основы ООП.

# Задание 1.1

Условие:

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";

*function* createNote(*ChildsArr*, *surname*, *age*)

{

*let* n = *ChildsArr*.length;

    for (*let* i = 0; i < n; i++)

    {

        if (*surname* === *ChildsArr*[i].surname)

        {

            console.log("Ребёнок с такой фамилией уже есть");

            return 0

        }

    }

*let* child = {surname : *surname*, age : *age*}

*ChildsArr*.push(child);

}

*function* delet(*ChildsArr*, *surname*)

{

*let* n = *ChildsArr*.length;

    for (*let* i = 0; i < n; i++)

    {

        if (*surname* === *ChildsArr*[i].surname)

        {

*ChildsArr*.splice(i, 1);

            break

        }

    }

}

*function* read(*ChildsArr*, *surname*)

{

*let* n = *ChildsArr*.length;

    for (*let* i = 0; i < n; i++)

    {

        if (*surname* === *ChildsArr*[i].surname)

        {

            console.log(*ChildsArr*[i]);

            break

        }

    }

}

*function* update(*ChildsArr*, *surname*, *newsurnmae*, *newage*)

{

*let* n = *ChildsArr*.length;

    for (*let* i = 0; i < n; i++)

    {

        if (*surname* === *ChildsArr*[i].surname)

        {

*ChildsArr*[i].surname = *newsurnmae*;

*ChildsArr*[i].age = *newage*;

            break

        }

    }

}

// Получение среднего возраста детей

*function* AverageAge(*ChildsArr*)

{

*let* n = *ChildsArr*.length;

*let* avg = 0;

    for (*let* i = 0; i < n; i++)

    {

        avg += *ChildsArr*[i].age;

    }

    avg = parseFloat(avg / n);

    return avg;

}

// информации о детях, возраст которых входит в заданный отрезок

*function* FindByAge(*ChildsArr*, *from*, *to*)

{

*let* n = *ChildsArr*.length;

*let* i = 0;

    console.log("Children with age from "+ *from* + " to " + *to*);

    while (i < n)

    {

        if (*from* <= *ChildsArr*[i].age && *ChildsArr*[i].age <= *to*)

        {

            console.log(*ChildsArr*[i]);

        }

        i++;

    }

    console.log();

}

// информации о детях, фамилия которых начинается с заданной буквы

*function* FindFirstLetter(*ChildsArr*, *letter*)

{

*let* n = *ChildsArr*.length;

*let* first

    console.log("Children with surname beginning = " + letter);

    for (*let* i = 0; i < n; i++)

    {

        first = *ChildsArr*[i].surname.charAt(0);

        if (first === *letter*)

        {

            console.log(*ChildsArr*[i]);

        }

    }

    console.log();

}

// информации о детях, фамилия которых длиннее заданного количества символов

*function* FindByLength(*ChildsArr*, *length*)

{

*let* n = *ChildsArr*.length;

*let* len;

    console.log("Children with surname > " + length);

    for (*let* i = 0; i < n; i++)

    {

        len = *ChildsArr*[i].surname.length;

        if (len > *length*)

        {

            console.log(*ChildsArr*[i]);

        }

    }

    console.log();

}

// информации о детях, фамилия которых начинается с гласной буквы

*function* FindByVowels(*ChildsArr*)

{

*let* vowels = ["а", "е", "и", "о", "у", "э", "я", "ю", "А", "Е", "И", "О", "У", "Э", "Я", "Ю"];

*let* n = *ChildsArr*.length;

*let* first;

    console.log("Children with surname started by vowel: ");

    for (*let* i = 0; i < n; i++)

    {

        first = *ChildsArr*[i].surname.charAt(0);

        for (*let* j = 0; j < vowels.length; j++)

        {

            if (first === vowels[j])

            {

                console.log(ChildsArr[i]);

                break

            }

        }

    }

    console.log();

}

// Страший ребёнок

*function* FindOld(*ChildsArr*)

{

*let* n = *ChildsArr*.length;

*let* old = *ChildsArr*[0];

*let* maxAge = *ChildsArr*[0].age;

    for (*let* i = 0; i < n; i++)

    {

        if (maxAge < *ChildsArr*[i].age)

        {

            maxAge = *ChildsArr*[i].age;

            old = *ChildsArr*[i];

        }

    }

    console.log("The oldest: ")

    console.log(old);

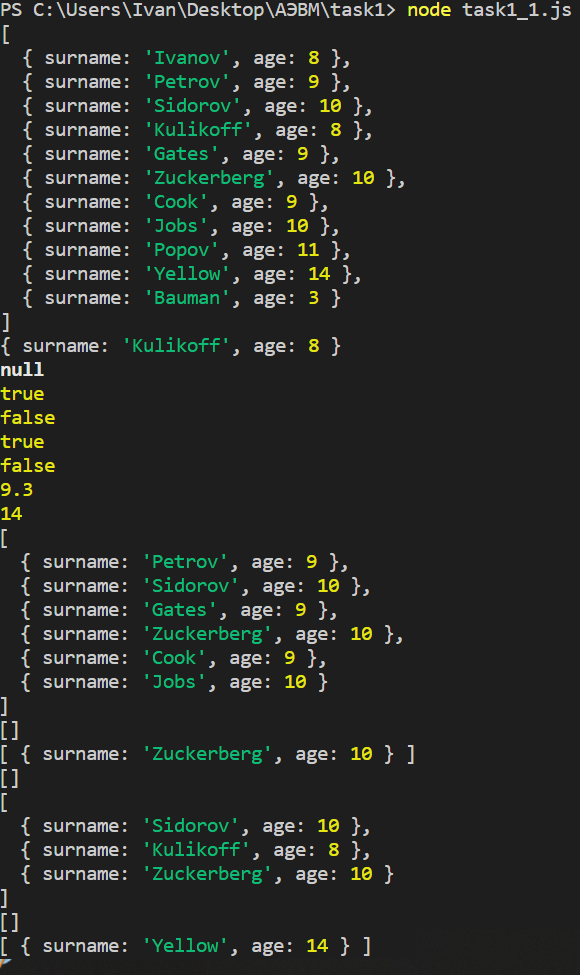
}

Набор тестов

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемые выходные данные | Реальные выходные данные |
| createNote(ChildsArr, "Лоев", 15); | true  Ребенок добавлен в массив | true |
| createNote(ChildsArr, "Лоев", 12); | false  Ребенок с неуникальной фамилией | false |
| read(ChildsArr, "Лоев"); | { surname: 'Лоев', age: 15 } | { surname: 'Лоев', age: 15 } |
| read(ChildsArr, "Лок"); | null | null |
| update(ChildsArr, "Лоев", "Лынев", 16); | true | true |
| delet(ChildsArr, "Рыбников"); | true | true |
| delet(ChildsArr, "Рыбников"); | false | false |
| AverageAge(ChildsArr) | 13.66 | 13.66 |
| FindOld(ChildsArr); | { surname: 'Лынев', age: 16 } | { surname: 'Лынев', age: 16 } |
| FindByAge(ChildsArr, 10, 14); | Children with age from 10 to 14  { surname: 'Иванов', age: 13 }  { surname: 'Нарица', age: 12 } | Children with age from 10 to 14  { surname: 'Иванов', age: 13 }  { surname: 'Нарица', age: 12 } |
| FindByAge(ChildsArr, 18, 20); | [] | [] |
| FindFirstLetter(ChildsArr, "Н"); | Children with surname beginning = Н  { surname: 'Нарица', age: 12 } | Children with surname beginning = Н  { surname: 'Нарица', age: 12 } |
| FindFirstLetter(ChildsArr, "Я"); | [] | [] |
| FindByLength(ChildsArr, 5); | Children with surname > 5  { surname: 'Иванов', age: 13 } | Children with surname > 5  { surname: 'Иванов', age: 13 } |
| FindByLength(ChildsArr, 20); | [] | [] |
| FindByVowels(ChildsArr); | Children with surname started by vowel:  { surname: 'Иванов', age: 13 } | Children with surname started by vowel:  { surname: 'Иванов', age: 13 } |

Все тесты пройдены успешно.

Пример работы:



# Задание 1.2

Условие:

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 на node.js

"use strict"

let arr = []

arr.push({group: "IU7-51B", studentCard: 1, marks: [5, 5, 5]});

arr.push({group: "IU7-51B", studentCard: 4, marks: [5, 4]});

arr.push({group: "IU7-51B", studentCard: 6, marks: [4]});

arr.push({group: "IU7-51B", studentCard: 2, marks: [3, 5, 5]});

arr.push({group: "IU7-52B", studentCard: 5, marks: [3, 3, 3, 3]});

arr.push({group: "IU7-52B", studentCard: 3, marks: [3]});

arr.push({group: "IU7-52B", studentCard: 9, marks: [5]});

function createStudent(group, studentCard, marks) {

    let student = {group: group, studentCard: studentCard, marks: marks};

    let f = true;

    let n = arr.length;

    for (let i = 0; i < n && f; i++)

        if (arr[i]["studentCard"] === student["studentCard"])

            f = false;

    if (f)

        arr.push(student);

return f;

}

function readStudent(value) {

    let n = arr.length;

    for (let i = 0; i < n; i++)

        if (arr[i]["studentCard"] === value)

            return arr[i];

    return null;

}

function updateStudent(key, index, new\_value) {

    if (index < 0 || index >= arr.length)

        return false;

    if (key != "group" && key != "studentCard" && key != "marks")

        return false;

    let f = true;

    if (key === "studentCard") {

        let n = arr.length;

        for (let i = 0; i < n; i++)

            if (arr[i]["studentCard"] === new\_value)

                f = false;

    }

    if (new\_value && f) {

        arr[index][key] = new\_value;

        return true;

    } else

        return false;

}

function deleteStudent(value) {

    let n = arr.length;

    let f = false;

    for (let i = 0; i < n && !f; i++) {

        if (arr[i]["studentCard"] == value) {

            arr.splice(i, 1);

            i--;

            n--;

            f = true;

        }

    }

    return f;

}

function getAvgMark(value) {

    let n = arr.length;

    for (let i = 0; i < n; i++)

        if (arr[i]["studentCard"] === value) {

            let sum = 0;

            let m = arr[i]["marks"].length;

            for (let j = 0; j < m; j++)

                sum += arr[i]["marks"][j];

            return sum / m;

        }

    return NaN;

}

function getGroup(group) {

    let res = [];

    let n = arr.length;

    for (let i = 0; i < n; i++)

        if (arr[i]["group"] === group)

            res.push(arr[i]);

    return res;

}

function getMaxMarkInGroup(group) {

    let g = getGroup(group);

    let n = g.length;

    let maxMarks = 0;

    let res = null;

    for (let i = 0; i < n; i++) {

        if (g[i]["marks"].length > maxMarks) {

            maxMarks = g[i]["marks"].length;

            res = g[i];

        }

    }

    return res;

}

function getNoMarksStudent() {

    let n = arr.length;

    for (let i = 0; i < n; i++)

        if (arr[i]["marks"].length === 0)

            return arr[i];

    return null;

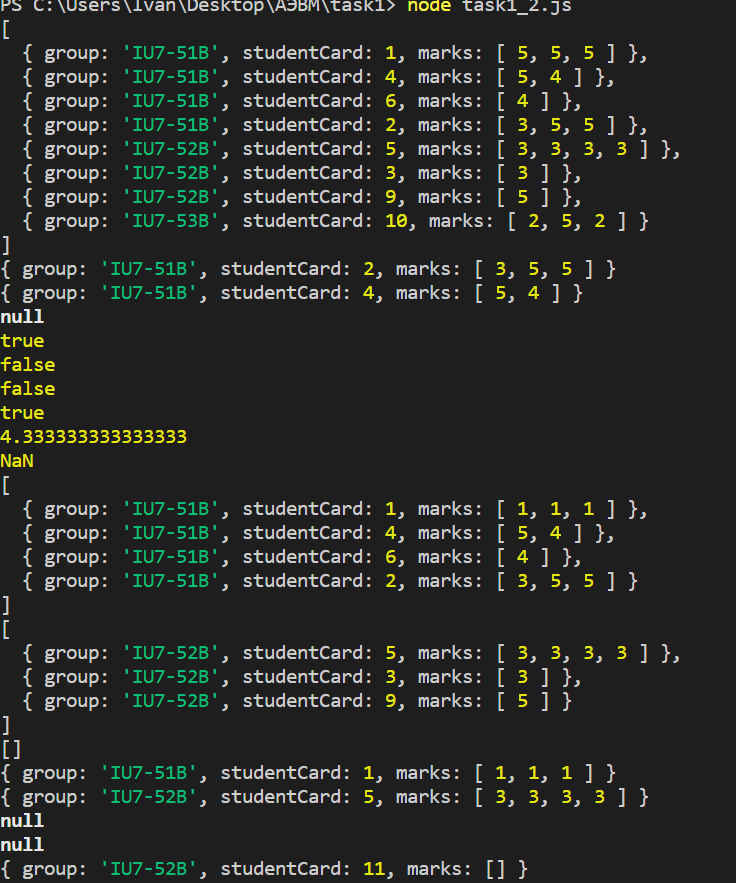
}

Набор тестов

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемые выходные данные | Реальные выходные данные |
| createStudent("IU7-53B", 1, [2, 2, 2]) | false | false |
| createStudent("IU7-53B", 10, [2, 5, 2]) | true | true |
| readStudent(2) | { group: 'IU7-51B', studentCard: 2, marks: [ 3, 5, 5 ] } | { group: 'IU7-51B', studentCard: 2, marks: [ 3, 5, 5 ] } |
| readStudent(4) | {group: 'IU7-51B', studentCard: 4, marks: [ 5, 4 ]} | {group: 'IU7-51B', studentCard: 4, marks: [ 5, 4 ]} |
| readStudent(42) | null | null |
| updateStudent("marks", 0, [1, 1, 1]) | true | true |
| updateStudent("studentCard", 0, 4) | false | false |
| deleteStudent(150) | false | false |
| deleteStudent(10) | true | true |
| getAvgMark(2) | 4.333 | 4.333 |
| getAvgMark(150) | NaN | NaN |
| getGroup("IU7-51B") | [{ group: 'IU7-51B', studentCard: 4, marks: [ 5, 4 ] },  { group: 'IU7-51B', studentCard: 6, marks: [ 4 ] },  { group: 'IU7-51B', studentCard: 2, marks: [ 3, 5, 5 ]}] | [{ group: 'IU7-51B', studentCard: 4, marks: [ 5, 4 ] },  { group: 'IU7-51B', studentCard: 6, marks: [ 4 ] },  { group: 'IU7-51B', studentCard: 2, marks: [ 3, 5, 5 ]}] |
| getGroup("IU7-52B") | [{ group: 'IU7-52B', studentCard: 5, marks: [ 3, 3, 3, 3 ] },  { group: 'IU7-52B', studentCard: 3, marks: [ 3 ] },  { group: 'IU7-52B', studentCard: 9, marks: [ 5 ] }] | [{ group: 'IU7-52B', studentCard: 5, marks: [ 3, 3, 3, 3 ] },  { group: 'IU7-52B', studentCard: 3, marks: [ 3 ] },  { group: 'IU7-52B', studentCard: 9, marks: [ 5 ] }] |
| getGroup("IU7-53B") | [] | [] |
| getMaxMarkInGroup("IU7-51B") | { group: 'IU7-51B', studentCard: 1, marks: [ 1, 1, 1 ] } | { group: 'IU7-51B', studentCard: 1, marks: [ 1, 1, 1 ] } |
| getMaxMarkInGroup("IU7-52B") | { group: 'IU7-52B', studentCard: 5, marks: [ 3, 3, 3, 3 ] } | { group: 'IU7-52B', studentCard: 5, marks: [ 3, 3, 3, 3 ] } |
| getMaxMarkInGroup("IU7-53B") | null | null |
| getNoMarksStudent() | null | null |
| createStudent("IU7-52B", 11, [])  getNoMarksStudent() | { group: 'IU7-52B', studentCard: 11, marks: [] } | { group: 'IU7-52B', studentCard: 11, marks: [] } |

Все тесты пройдены успешно.

Пример работы



# Задание 1.3

Условие:

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 программы на node.js

"use strict"

let arr = []

arr.push({name: "O", x: 0, y: 0});

arr.push({name: "X", x: 1, y: 0});

arr.push({name: "Y", x: 0, y: 1});

arr.push({name: "z", x: -0.25, y: -0.25})

arr.push({name: "R", x: 1, y: 1});

arr.push({name: "W", x: 4, y: 2.5});

function createPoint(name, x, y) {

    let point = {name: name, x: x, y: y};

    let f = true;

    let n = arr.length;

    for (let i = 0; i < n && f; i++)

        if (arr[i]["name"] === point["name"])

            f = false;

    if (f)

        arr.push(point);

    return f;

}

function readPoint(value) {

    let n = arr.length;

    for (let i = 0; i < n; i++)

        if (arr[i]["name"] === value)

            return arr[i];

    return null;

}

function updatePoint(key, index, new\_value) {

    if (index < 0 || index >= arr.length)

        return false;

    if (key != "name" && key != "x" && key != "y")

        return false;

    let f = true;

    if (key === "name") {

        let n = arr.length;

        for (let i = 0; i < n; i++)

            if (arr[i]["name"] === new\_value)

                f = false;

    }

    if (new\_value && f) {

        arr[index][key] = new\_value;

        return true;

    } else

        return false;

}

function deletePoint(value) {

    let n = arr.length;

    let f = false;

    for (let i = 0; i < n && !f; i++) {

        if (arr[i]["name"] === value) {

            arr.splice(i, 1);

            i--;

            n--;

            f = true;

        }

    }

    return f;

}

function getDistance(p1, p2) {

    return Math.sqrt((p2.x - p1.x) \* (p2.x - p1.x) + (p2.y - p1.y) \* (p2.y - p1.y));

}

function sub(p1, p2) {

    return {name: p1.name + p2.name, x: p2.x - p1.x, y: p2.y - p1.y};

}

function greatestDistance() {

    let maxDist = 0;

    let n = arr.length;

    for (let i = 0; i < n; i++)

        for (let j = 0; j < n; j++)

            if (i != j) {

                let dist = getDistance(arr[i], arr[j]);

                if (dist > maxDist)

                    maxDist = dist;

            }

    return maxDist;

}

function notFutherFrom(point, dist) {

    let res = [];

    let n = arr.length;

    for (let i = 0; i < n; i++) {

        if (getDistance(point, arr[i]) <= dist)

            res.push(arr[i]);

    }

    return res;

}

function len(v1) {

    return Math.sqrt(v1.x \* v1.x + v1.y \* v1.y);

}

function getCosAngle(v1, v2) {

    return (v1.x \* v2.x + v1.y \* v2.y) / len(v1) / len(v2);

}

function onOneSideOf(direction) {

    let res = [];

    let n = arr.length;

    for (let i = 0; i < n; i++) {

        if (getCosAngle(direction, arr[i]) > 0)

            res.push(arr[i]);

    }

    return res;

}

function insideRect(ld, ru) {

    let res = [];

    let n = arr.length;

    for (let i = 0; i < n; i++) {

        let v1 = sub(ld, arr[i]);

        let v2 = sub(ru, arr[i]);

        if (v1.x > 0 && v1.y > 0 && v2.x < 0 && v2.y < 0)

            res.push(arr[i]);

    }

    return res;

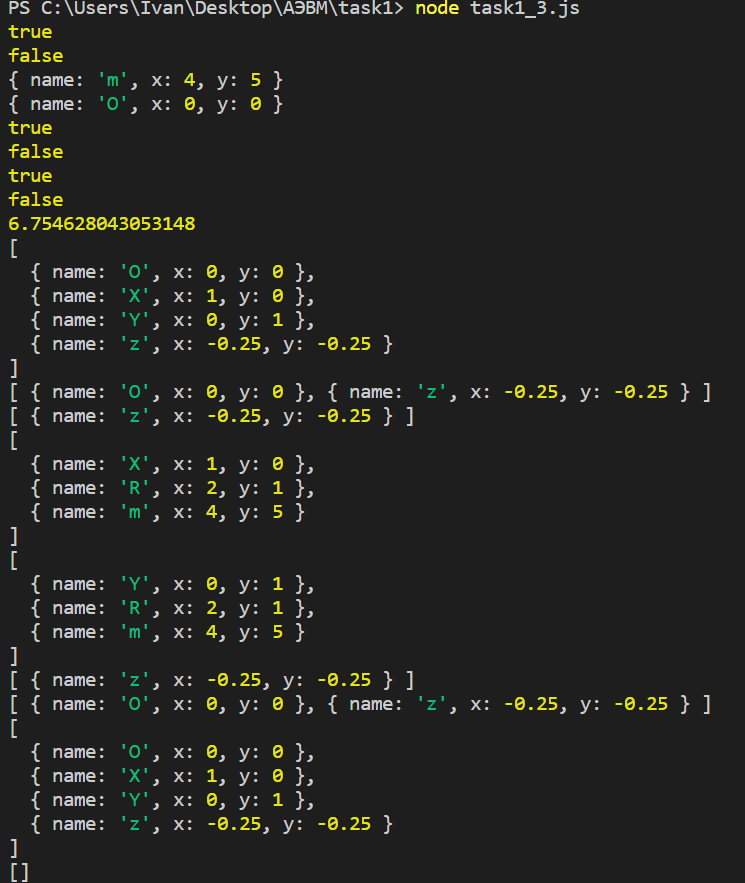
}

Набор тестов

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемые выходные данные | Реальные выходные данные |
| createPoint("m", 4, 5) | true | true |
| createPoint("O", 1, 1) | false | false |
| readPoint("m") | { name: 'm', x: 4, y: 5 } | { name: 'm', x: 4, y: 5 } |
| readPoint("O") | { name: 'O', x: 0, y: 0 } | { name: 'O', x: 0, y: 0 } |
| updatePoint("x", 3, 2) | true | true |
| updatePoint("name", 4, "Y") | false | false |
| deletePoint("W") | true | true |
| deletePoint("Hello") | false | false |
| greatestDistance() | 6.754 | 6.754 |
| notFutherFrom(  {name: "O", x: 0, y: 0}, 1) | [{ name: 'O', x: 0, y: 0 },  { name: 'X', x: 1, y: 0 },  { name: 'Y', x: 0, y: 1 },  { name: 'z', x: -0.25, y: -0.25 }] | [{ name: 'O', x: 0, y: 0 },  { name: 'X', x: 1, y: 0 },  { name: 'Y', x: 0, y: 1 },  { name: 'z', x: -0.25, y: -0.25 }] |
| notFutherFrom(  {name: "O", x: 0, y: 0}, 0.5) | [ { name: 'O', x: 0, y: 0 },  { name: 'z', x: -0.25, y: -0.25 } ] | [ { name: 'O', x: 0, y: 0 }, { name: 'z', x: -0.25, y: -0.25 } ] |
| onOneSideOf(  {name: "left", x: -1, y: 0}) | [ { name: 'z', x: -0.25, y: -0.25 } ] | [ { name: 'z', x: -0.25, y: -0.25 } ] |
| onOneSideOf(  {name: "right", x: 1, y: 0}) | [{ name: 'X', x: 1, y: 0 },  { name: 'R', x: 2, y: 1 },  { name: 'm', x: 4, y: 5 }] | [{ name: 'X', x: 1, y: 0 },  { name: 'R', x: 2, y: 1 },  { name: 'm', x: 4, y: 5 }] |
| onOneSideOf(  {name: "up", x: 0, y: 1}) | [{ name: 'Y', x: 0, y: 1 },  { name: 'R', x: 2, y: 1 },  { name: 'm', x: 4, y: 5 }] | [{ name: 'Y', x: 0, y: 1 },  { name: 'R', x: 2, y: 1 },  { name: 'm', x: 4, y: 5 }] |
| onOneSideOf(  {name: "down", x: 0, y: -1}) | [ { name: 'z', x: -0.25, y: -0.25 } ] | [ { name: 'z', x: -0.25, y: -0.25 } ] |
| insideRect({name: "left-down", x: -0.5, y: -0.5}, {name: "right-up", x: 1, y: 1}) | [ { name: 'O', x: 0, y: 0 }, { name: 'z', x: -0.25, y: -0.25 } ] | [ { name: 'O', x: 0, y: 0 }, { name: 'z', x: -0.25, y: -0.25 } ] |
| insideRect({name: "left-down", x: -0.5, y: -0.5}, {name: "right-up", x: 1.01, y: 1.01}) | [{ name: 'O', x: 0, y: 0 },  { name: 'X', x: 1, y: 0 },  { name: 'Y', x: 0, y: 1 },  { name: 'z', x: -0.25, y: -0.25 }] | [{ name: 'O', x: 0, y: 0 },  { name: 'X', x: 1, y: 0 },  { name: 'Y', x: 0, y: 1 },  { name: 'z', x: -0.25, y: -0.25 }] |
| insideRect({name: "left-down", x: -0.2, y: -0.2}, {name: "right-up", x: -0.1, y: -0.1}) | [] | [] |

Все тесты пройдены успешно.

Пример работы



# Задание 2.1

Условие:

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 на языке node.js

class Point {

    constructor(x, y) {

        this.x = x;

        this.y = y;

    }

    print() {

        console.log("(" + this.x + ' ' + this.y + ")");

    }

}

class Line {

    constructor(a, b) {

        this.a = a;

        this.b = b;

    }

    print() {

        console.log("Begin point: ");

        this.a.print();

        console.log("End point: ");

        this.b.print();

    }

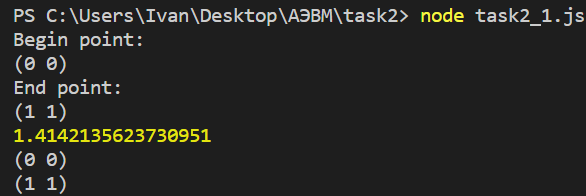
}

Набор тестов

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемые выходные данные | Реальные выходные данные |
| a = new Line(new Point(0, 0),  new Point(1, 1));  a.print(); | Begin point: (0 0)  End point: (1 1) | Begin point:  (0 0)  End point:  (1 1) |
| a.getLength() | 1.414 | 1.414 |
| a.a.print() | (0 0) | (0 0) |
| a.b.print() | (1 1) | (1 1) |

Все тесты пройдены успешно.

Пример работы



# Задание 2.2

Условие:

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 программы на языке node.js

class Triangle {

    constructor(a, b, c) {

        this.a = a;

        this.b = b;

        this.c = c;

    }

    isExists() {

        return this.a < this.b + this.c && this.b < this.a + this.c && this.c < this.a + this.b;

    }

    perimeter() {

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

    }

    area() {

        let p = this.perimeter() / 2;

        return Math.sqrt(p \* (p - this.a) \* (p - this.b) \* (p - this.c));

    }

    isRectangular() {

        return (this.a \* this.a + this.b \* this.b == this.c \* this.c ||

                this.a \* this.a + this.c \* this.c == this.b \* this.b ||

                this.b \* this.b + this.c \* this.c == this.a \* this.a)

    }

}

Набор тестов

tr = new Triangle(1, 1, Math.sqrt(2))

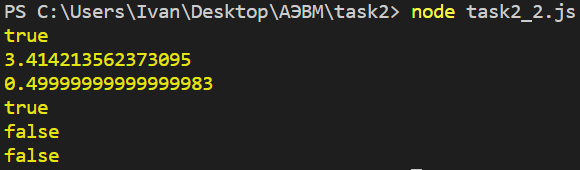
tr1 = new Triangle(5, 1, 1)

tr2 = new Triangle(1, 2, 2.5)

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемые выходные данные | Реальные выходные данные |
| tr.isExists() | true | True |
| tr1.isExists() | false | false |
| tr.perimeter() | 3.414 | 3.414 |
| tr.area() | 0.5 | 0.499 |
| tr.isRectangular() | true | true |
| tr2.isRectangular() | false | false |

Все тесты пройдены успешно.

Пример работы



# Задание 2.3

Условие:

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 программы на языке node.js

for (let i = 0; i < 11; i++)

    setTimeout(() => { console.log(i); }, 2000 \* i);

for (let i = 11; i < 21; i++)

    setTimeout(() => { console.log(i); }, 22000 + (i - 11) \* 1000);

setInterval( () => {

    for (let i = 0; i < 11; i++)

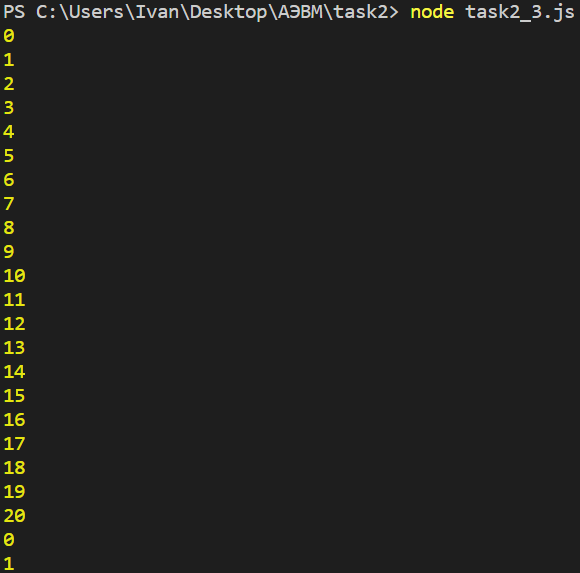
        setTimeout(() => { console.log(i); }, 2000 \* i);

    for (let i = 11; i < 21; i++)

        setTimeout(() => { console.log(i); }, 22000 + (i - 11) \* 1000);

}, 33000)

Пример работы



Вывод

В результате лабораторной работы, я познакомился в основами синтаксиса javascript, фреймворком node.js, изучил стрелочные функции и основы ООП.