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

Тема Основы JS.

Студент Воякин А. Я.

Группа ИУ7-54Б

Оценка (баллы) _____

Преподаватель Попов А. Ю.

Москва.
2020 г.

Условие задания

Задание 1.1

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

Задание 1.2

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

Задание 1.3

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

Задание 2.1

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 также метод получения длины отрезка.

Задание 2.2

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

Задание 2.3

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.

Задание 1.1

Листинг программы:

```
"use strict";

let children = {
    data: [],

    // Функция создания объекта child и добавление его в хранилище.
    create: function(surname, age) {
        // Проверка на корректность введённого возраста.
        if (age <= 0) {
            console.log("Возраст должен быть положильным целым числом");
            return;
        }

        // Проверка на уникальность фамилии.
        if (this.data.find(item => item.surname.toLowerCase() === surname.toLowerCase()) !==
undefined) {
            console.log("Ребёнок с фамилией", '"'+surname+'"', "уже присутствует в хранилище.");
            return;
        }

        this.data.push({
            "surname": surname,
            "age": age
        });
    },

    // Функция вывода возраста ребенка по фамилии.
    read: function(surname) {
        // Проверка наличия ребенка с необходимой фамилией в хранилище.
        let child = this.data.find(item => item.surname.toLowerCase() === surname.toLowerCase());
        if (child === undefined) {
            console.log("Ребёнок с фамилией", '"'+surname+'"', "не найден.");
            return;
        }

        return(child);
    },

    // Функция модифицирования информации о ребёнке с введённой фамилией.
    update: function(surname old, surname, age) {
        // Проверка на наличие ребенка для уделания в хранилище.
        let ind = this.data.findIndex(item => item.surname.toLowerCase() ===
surname_old.toLowerCase());
        if (ind === -1) {
            console.log("Ребенок с фамилией", '"'+surname_old+'"', "не найден.");
            return;
        }

        // Проверка на уникальность фамилии добавляемого ребёнка.
        let child = this.data.find(item => item.surname.toLowerCase() === surname.toLowerCase());
        if (child !== undefined && surname old.toLowerCase() !== surname.toLowerCase()) {
            console.log("Ребенок с фамилией", '"'+surname+'"', "уже присутствует в хранилище.")
            return;
        }

        // Обновление информации о ребёнке.
        this.data.splice(ind, 1, {"surname": surname, "age": age });
    },

    // Функция удаления информации о ребёнке по его фамилии.
    delete: function(surname) {
        // Проверка на наличие ребенка для уделания в хранилище.
        let ind = this.data.findIndex(item => item.surname.toLowerCase() ===
surname.toLowerCase());
        if (ind === -1) {
            console.log("Ребенок с фамилией", '"'+surname+'"', "не найден.");
            return;
        }

        // Удаление информации о ребёнке.
        this.data.splice(ind, 1);
    },

    // Фунция вывода хранилища.
    output: function() {
```

```
        console.log("\n---------ХРАНИЛИЩЕ---------");
        for (let i = 0; i < this.data.length; i++) {
            console.log(this.data[i].surname, this.data[i].age);
        }
        console.log("--------------------------");
    },

    // Получение среднего возраста детей.
    average_age: function() {
        // Проверка на наличие информации о детях в хранилище.
        if (this.data.length === 0) {
            console.log("Пустое хранилище информации о детях.");
            return undefined;
        }

        // Подсчёт среднего возрасте детей в хранилище.
        let sum = 0;
        for (let i = 0; i < this.data.length; i++) {
            sum += this.data[i].age;
        }
        sum /= this.data.length;
        return sum;
    },

    // Получение информации о самом старшем ребёнке.
    find_oldest: function() {
        // Проверка на наличие информации о детях в хранилище.
        if (this.data.length === 0) {
            console.log("Пустое хранилище информации о детях.");
            return undefined;
        }

        // Нахождение максимального возраста детей в хранилище.
        let max_age = 0;
        for (let i = 0; i < this.data.length; i++) if (this.data[i].age > max_age){
            max_age = this.data[i].age;
        }

        // Нахождение всех детей с максимальным возрастом из хранилища.
        return this.data.filter(item => item.age === max_age);
    },

    // Получение информации о детях, возраст которых входит в заданный отрезок.
    in_age_range: function(from, to) {
        // Проверка на наличие информации о детях в хранилище.
        if (this.data.length === 0) {
            console.log("Пустое хранилище информации о детях.");
            return undefined;
        }

        // Фильтрация.
        return this.data.filter(item => item.age >= from & item.age <= to);
    },

    // Получение информации о детях, фамилия которых начинается с заданной буквы.
    starts_with_letter: function(letter) {
        // Проверка на наличие информации о детях в хранилище.
        if (this.data.length === 0) {
            console.log("Пустое хранилище информации о детях.");
            return undefined;
        }

        // Фильтрация.
        return this.data.filter(item => item.surname[0].toLowerCase() === letter.toLowerCase());
    },

    // Получение информации о детях, фамилия которых длиннее заданного количества символов.
    sur_longer_then_num: function(num) {
        // Проверка на наличие информации о детях в хранилище.
        if (this.data.length === 0) {
            console.log("Пустое хранилище информации о детях.");
            return undefined;
        }

        // Фильтрация.
        return this.data.filter(item => item.surname.length > num);
    },

    // Получение информации о детях, фамилия которых начинается с гласной буквы.
    sur_starts_with_vowel(){
        // Проверка на наличие информации о детях в хранилище.
```

```
        if (this.data.length === 0) {
            console.log("Пустое хранилище информации о детях.");
            return undefined;
        }

        // Фильтрация.
        let vowels = "аеіоуыэяюёиоуыэяюя";
        return this.data.filter(item => vowels.search(item.surname[0].toLowerCase()) !== -1);
    }
}

// Create test
console.log("\n--------CREATE TEST--------");
children.create("Воякин", 10);
children.create("Пиксаев", 11);
children.create("Вильданов", 6);
children.create("Дорогова", 13);
children.create("Капичникова", 1);
children.create("Сорокина", 16);
// Negative test
children.create("пиксаев", 12);
children.output();

// Read test
console.log("\n---------READ TEST---------");
console.log(children.read("Воякин"));
// Negative test
console.log(children.read("Дорогова"));

// Update test
console.log("\n--------UPDATE TEST--------");
children.update("Вильданов", "Попов", 3);
children.update("Воякин", "Воякин", 12);
// Negative test
children.update("Пиксаев", "Воякин", 12);
children.update("Тюрин", "Попов", 13);
children.output();

// Delete test
console.log("\n--------DELETE TEST--------");
children.delete("Попов");
// Negative test
children.delete("Вильданов");
children.output();

// Average age test
console.log("\n------AVERAGE AGE TEST-----");
console.log("Средний возраст детей в хранилище: ", children.average_age());

// Find oldest child test
console.log("\n------FIND_OLDEST TEST-----");
console.log(children.find_oldest());

// In age range test
console.log("\n----------IN_AGE TEST-------");
console.log(children.in_age_range(10, 11));

// Surname starts from current letter test
console.log("\n--STARTS WITH LETTER TEST--");
console.log(children.starts_with_letter('в'));

// Surname longer then number
console.log("\n----LONGER THEN NUM TEST---");
console.log(children.sur longer then num(7));

// Surname starts with vowel
console.log("\n---STARTS_WITH_VOWEL TEST--");
children.create("Орехов", 3);
console.log(children.sur_starts_with_vowel());
```

Результаты выполнения тестов:

```
--------CREATE TEST--------
Ребёнок с фамилией "пиксаев" уже присутствует в хранилище.

---------ХРАНИЛИЩЕ---------
Воякин 10
Пиксаев 11
Вильданов 6
```

```
Дорогова 13
Капичникова 1
Сорокина 16
---------------------------

---------READ TEST---------
{ surname: 'Воякин', age: 10 }
{ surname: 'Дорогова', age: 13 }

--------UPDATE TEST--------
Ребенок с фамилией "Воякин" уже присутствует в хранилище.
Ребенок с фамилией "Тюрин" не найден.

---------ХРАНИЛИЩЕ---------
Воякин 12
Пиксаев 11
Попов 3
Дорогова 13
Капичникова 1
Сорокина 16
---------------------------

--------DELETE TEST--------
Ребенок с фамилией "Вильданов" не найден.

---------ХРАНИЛИЩЕ---------
Воякин 12
Пиксаев 11
Дорогова 13
Капичникова 1
Сорокина 16
---------------------------

------AVERAGE AGE TEST-----
Средний возраст детей в хранилище:  10.6

------FIND_OLDEST TEST-----
[ { surname: 'Сорокина', age: 16 } ]

---------IN_AGE TEST-------
[ { surname: 'Пиксаев', age: 11 } ]

--STARTS WITH LETTER TEST--
[ { surname: 'Воякин', age: 12 } ]

----LONGER THEN NUM TEST---
[
  { surname: 'Дорогова', age: 13 },
  { surname: 'Капичникова', age: 1 },
  { surname: 'Сорокина', age: 16 }
]

---STARTS_WITH_VOWEL TEST--
[ { surname: 'Орехов', age: 3 } ]
```

Задание 1.2

Листинг программы:

```
"use strict";

let students = {
    data: [],

    // Функция создания объекта student и добавление его в хранилище.
    create: function(group, studentID, grades) {
        // Проверка на уникальность номера студенческого билета.
        if (this.data.find(item => item.studentID === studentID) !== undefined) {
            console.log("Студент со студенческим билетом №", studentID, "уже присутствует в
хранилище.");
            return;
        }

        this.data.push({
            "group": group,
            "studentID": studentID,
            "grades": grades
        });
    },

    // Функция вывода информации о студенте по номеру его студенческого билета.
    read: function(studentID) {
        // Проверка наличия студента с нужным номером студенческого билета в хранилище.
        let student = this.data.find(item => item.studentID === studentID);
        if (student === undefined) {
            console.log("Студент со студенческим билетом №", studentID, "не найден.");
            return;
        }

        return student;
    },

    // Функция модифицирования информации о студенте с введённым номером студ. билета.
    update: function(studentID_old, group, studentID, grades) {
        // Проверка на наличие студента для уделания в хранилище.
        let ind = this.data.findIndex(item => item.studentID === studentID_old);
        if (ind === -1) {
            console.log("Студент со студенческим билетом №", studentID_old, "не найден.");
            return;
        }

        // Проверка на уникальность номера студенческого билета добавляемого студента.
        let student = this.data.find(item => item.studentID === studentID);
        if (student !== undefined & studentID_old !== studentID) {
            console.log("Студент со студенческим билетом №", studentID, "уже присутствует в
хранилище.");
            return;
        }

        // Обновление информации о студенте.
        this.data.splice(ind, 1, {"group": group, "studentID": studentID, "grades": grades});
    },

    // Функция удаления информации о студенте по номеру его студенческого билета.
    delete: function(studentID) {
        // Проверка на наличие студента для уделания в хранилище.
        let ind = this.data.findIndex(item => item.studentID === studentID);
        if (ind === -1) {
            console.log("Студент со студенческим билетом №", studentID, "не найден.");
            return;
        }

        // Удаление информации о студенте.
        this.data.splice(ind, 1);
    },

    // Фунция вывода хранилища.
    output: function() {
        console.log("\n---------ХРАНИЛИЩЕ---------");
        for (let i = 0; i < this.data.length; i++) {
            console.log(this.data[i].group, this.data[i].studentID, this.data[i].grades);
        }
        console.log("--------------------------\n");
```

```
    },

    // Получение средней оценки заданного студента.
    average_grade: function(studentID) {
        // Проверка наличия информации о студенте в хранилище.
        let ind = this.data.findIndex(item => item.studentID === studentID);
        if (ind === -1) {
            console.log("Студент со студенческим билетом №", studentID, "не найден.");
            return undefined;
        }

        // Проверка наличия оценок у заданного студента.
        let count = this.data[ind].grades.length;
        if (count === 0) {
            console.log("Студент со студенческим билетом №", studentID, "не имеет оценок.");
            return undefined;
        }

        // Подсчёт средней оценки заданного студента.
        let av_gr = 0;
        this.data[ind].grades.forEach(item => av_gr += item);
        return av_gr / count;
    },

    // Получение информации о студентах в заданной группе.
    in_group: function(group) {
        let result = this.data.filter(item => item.group === group);
        if (result.length === 0) {
            console.log("В группе", '"'+group+'"', "нет студентов.");
        }
        return result;
    },

    // Получение студента, у которого наибольшее количество оценок в заданной группе.
    max_grades_num_in_group: function(group) {
        // Фильтр по нужной группе.
        let cur_group = this.data.filter(item => item.group === group);

        // Проверка на наличие студентов в заданной группе.
        if (cur_group.length === 0) {
            console.log("В группе", '"'+group+'"', "нет студентов.");
            return [];
        }

        // Вычисление максимального кол-ва оценок.
        let max_num = 0;
        for (let i = 0; i < cur_group.length; i++) if (cur_group[i].grades.length > max_num) {
            max_num = cur_group[i].grades.length;
        }

        // Фильтр по нужному кол-ву оценок.
        let result = cur_group.filter(item => item.grades.length === max_num);
        return result;
    },

    // Получение студента, у которого нет оценок.
    empty_grades: function() {
        let result = this.data.filter(item => item.grades.length === 0);
        if (result.length === 0) {
            console.log("В хранилище отсутствуют студенты без оценок.");
        }
        return result;
    }
}

// Create test
console.log("\n--------CREATE TEST--------");
students.create("ИУ7-54Б", 192, [5, 4, 3, 5]);
students.create("ИУ7-54Б", 391, [5, 5]);
students.create("ИУ7-54Б", 474, [3, 3, 2]);
students.create("ИУ7-51Б", 701, [5, 5, 5, 5]);
students.create("ИУ7-51Б", 652, []);
students.create("ИУ7-51Б", 918, [2, 3, 2]);
// Negative test
students.create("ИУ7-54Б", 192, []);
students.output();

// Read test
console.log("\n---------READ TEST---------");
console.log(students.read(192));
console.log(students.read(652));
```

```
// Negative test
console.log(students.read(111));

// Update test
console.log("\n--------UPDATE TEST--------");
students.update(391, "ИУ7-53Б", 111, [4, 4, 3, 3, 4]);
students.update(918, "ИУ7-51Б", 918, [2, 3, 2, 4]);
// Negative test
students.update(999, "ИУ7-53Б", 888, [4, 3, 3, 3]);
students.update(192, "ИУ7-53Б", 918, [4, 3, 3, 3]);
students.output();

// Delete test
console.log("\n--------DELETE TEST--------");
students.delete(111);
// Negative test
students.delete(999);
students.output();

// Average grade test
console.log("\n----AVERAGE_GRADE TEST-----");
console.log("192: ", students.average_grade(192));
console.log("918: ", students.average_grade(918));
// Negative test
console.log("652: ", students.average_grade(652));
console.log("999: ", students.average_grade(999));

// In_group test
console.log("\n-------IN_GROUP TEST-------");
console.log(students.in group("ИУ7-54Б"));
// Negative test
console.log(students.in_group("ИУ7-52Б"));

// Max grades num in group
console.log("\n-MAX GRADES NUM IN GROUP TEST-");
console.log(students.max grades num in group("ИУ7-51Б"));
// Negative test
console.log(students.max_grades_num_in_group("ИУ7-55Б"));

// Empty_grades test
console.log("\n-----EMPTY_GRADES TEST-----");
console.log(students.empty_grades());
```

Результаты выполнения тестов:

```
--------CREATE TEST--------
Студент со студенческим билетом № 192 уже присутствует в хранилище.

---------ХРАНИЛИЩЕ---------
ИУ7-54Б 192 [ 5, 4, 3, 5 ]
ИУ7-54Б 391 [ 5, 5 ]
ИУ7-54Б 474 [ 3, 3, 2 ]
ИУ7-51Б 701 [ 5, 5, 5, 5 ]
ИУ7-51Б 652 []
ИУ7-51Б 918 [ 2, 3, 2 ]
---------------------------


---------READ TEST---------
{ group: 'ИУ7-54Б', studentID: 192, grades: [ 5, 4, 3, 5 ] }
{ group: 'ИУ7-51Б', studentID: 652, grades: [] }
Студент со студенческим билетом № 111 не найден.
undefined

--------UPDATE TEST--------
Студент со студенческим билетом № 999 не найден.
Студент со студенческим билетом № 918 уже присутствует в хранилище.

---------ХРАНИЛИЩЕ---------
ИУ7-54Б 192 [ 5, 4, 3, 5 ]
ИУ7-53Б 111 [ 4, 4, 3, 3, 4 ]
ИУ7-54Б 474 [ 3, 3, 2 ]
ИУ7-51Б 701 [ 5, 5, 5, 5 ]
ИУ7-51Б 652 []
ИУ7-51Б 918 [ 2, 3, 2, 4 ]
---------------------------
```

```
--------DELETE TEST--------
Студент со студенческим билетом № 999 не найден.

---------ХРАНИЛИЩЕ---------
ИУ7-54Б 192 [ 5, 4, 3, 5 ]
ИУ7-54Б 474 [ 3, 3, 2 ]
ИУ7-51Б 701 [ 5, 5, 5, 5 ]
ИУ7-51Б 652 []
ИУ7-51Б 918 [ 2, 3, 2, 4 ]
---------------------------


----AVERAGE GRADE TEST-----
192:  4.25
918:  2.75
Студент со студенческим билетом № 652 не имеет оценок.
652:  undefined
Студент со студенческим билетом № 999 не найден.
999:  undefined

-------IN_GROUP TEST-------
[
  { group: 'ИУ7-54Б', studentID: 192, grades: [ 5, 4, 3, 5 ] },
  { group: 'ИУ7-54Б', studentID: 474, grades: [ 3, 3, 2 ] }
]
В группе "ИУ7-52Б" нет студентов.
[]

-MAX GRADES NUM IN GROUP TEST-
[
  { group: 'ИУ7-51Б', studentID: 701, grades: [ 5, 5, 5, 5 ] },
  { group: 'ИУ7-51Б', studentID: 918, grades: [ 2, 3, 2, 4 ] }
]
В группе "ИУ7-55Б" нет студентов.
[]

-----EMPTY_GRADES TEST-----
[ { group: 'ИУ7-51Б', studentID: 652, grades: [] } ]
```

Задание 1.3

Листинг программы:

```
"use strict";

let nodes = {
    data: [],

    // Функция создания объекта node и добавление его в хранилище.
    create: function(name, x, y) {
        // Проверка на уникальность имени точки.
        if (this.data.find(item => item.name === name) !== undefined) {
            console.log("Точка с именем", '"'+name+'"' , "уже присутствует в хранилище.");
            return;
        }

        this.data.push({
            "name": name,
            "x": x,
            "y": y
        });
    },

    // Функция вывода информации о точке по её имени.
    read: function(name) {
        // Проверка наличия точки с нужным именем в хранилище.
        let node = this.data.find(item => item.name === name);
        if (node === undefined) {
            console.log("Точка с именем", '"'+name+'"', "не найдена.");
            return;
        }

        return node;
    },

    // Функция модифицирования информации о точке с нужным именем.
    update: function(name_old, name, x, y) {
        // Проверка на наличие точки для уделания в хранилище.
        let ind = this.data.findIndex(item => item.name === name_old);
        if (ind === -1) {
            console.log("Точка с названием", '"'+name_old+'"', "не найдена.");
            return;
        }

        // Проверка на уникальность имени добавляемой точки.
        let node = this.data.find(item => item.name === name);
        if (node !== undefined & name_old !== name) {
            console.log("Точка на именем", '"'+name+'"', "уже присутствует в хранилище.");
            return;
        }

        // Обновление информации о точке.
        this.data.splice(ind, 1, {"name": name, "x": x, "y": y});
    },

    // Функция удаления информации о точке по её имени.
    delete: function(name) {
        // Проверка на наличие точки для уделания в хранилище.
        let ind = this.data.findIndex(item => item.name === name);
        if (ind === -1) {
            console.log("Точка с именем", '"'+name+'"', "не найдена.");
            return;
        }

        // Удаление информации о точке.
        this.data.splice(ind, 1);
    },

    // Фунция вывода хранилища.
    output: function() {
        console.log("\n---------ХРАНИЛИЩЕ---------");
        for (let i = 0; i < this.data.length; i++) {
            console.log(this.data[i].name, this.data[i].x, this.data[i].y);
        }
        console.log("--------------------------\n");
    },
```

```
    // Получение двух точек, между которыми наибольшее расстояние
    max_distance: function() {
        if (this.data.length < 2) {
            console.log("Недостаточно точек определения расстояния.");
            return undefined;
        }
        let max_dis = 0;
        let i_buff = 0;
        let j_buff = 0;
        for (let i = 0; i < this.data.length - 1; i++) {
            let xi = this.data[i].x;
            let yi = this.data[i].y;
            for (let j = i + 1; j < this.data.length; j++) {
                let xj = this.data[j].x;
                let yj = this.data[j].y;
                let dis = Math.pow(Math.pow(xi - xj, 2) + Math.pow(yi - yj, 2), 1/2);
                if(dis > max_dis) {
                    i_buff = i;
                    j_buff = j;
                    max_dis = dis;
                }
            }
        }
        return([this.data[i_buff], this.data[j_buff]]);
    },

    // Получение точек, находящихся от заданной точки на расстоянии, не превышающем заданную
константу.
    in_range: function(x, y, dis) {
        let result = [];
        for (let i = 0; i < this.data.length; i++) {
            let xi = this.data[i].x;
            let yi = this.data[i].y;
            if (Math.pow(Math.pow(xi - x, 2) + Math.pow(yi - y, 2), 1/2) <= dis) {
                result.push(this.data[i]);
            }
        }
        return result;
    },

    // Получение точек, находящихся правее заданной оси координат.
    x_r: function(x) {
        return(this.data.filter(item => item.x > x));
    },

    // Получение точек, находящихся левее заданной оси координат.
    x_l: function(x) {
        return(this.data.filter(item => item.x < x));
    },

    // Получение точек, находящихся выше заданной оси координат.
    y_u: function(y) {
        return(this.data.filter(item => item.y > y));
    },

    // Получение точек, находящихся ниже заданной оси координат.
    y_d: function(y) {
        return(this.data.filter(item => item.y < y));
    },

    // Получение точек, входящих внутрь заданной прямоугольной зоны.
    in_rect: function(x1, y1, x2, y2) {
        let x11 = Math.min(x1, x2);
        let x22 = Math.max(x1, x2);
        let y11 = Math.min(y1, y2);
        let y22 = Math.max(y1, y2);
        return(this.data.filter(item => item.x > x11 & item.y > y11 & item.x < x22 & item.y <
y22));
    }
}

// Create test
console.log("\n--------CREATE TEST--------");
nodes.create("A", 0, 0);
nodes.create("B", 4, 4);
nodes.create("C", 0, 4);
nodes.create("D", -4, -4);
nodes.create("E", 0, -4);
// Negative test
nodes.create("C", 0, 0);
nodes.output();
```

```
// Read test
console.log("\n---------READ TEST---------");
console.log(nodes.read("A"));
console.log(nodes.read("D"));
// Negative test
console.log(nodes.read("F"));

// Update test
console.log("\n--------UPDATE TEST--------");
nodes.update("A", "A", 0, -1);
// Negative test
nodes.update("F", "G", 0, 0);
nodes.update("A", "B", 0, 0);
nodes.output();

// Delete test
console.log("\n--------DELETE TEST--------");
nodes.delete("A");
// Negative test
nodes.delete("F");
nodes.output();

// Max_distance test
console.log("\n-----MAX_DISTANCE TEST-----");
console.log(nodes.max_distance());

// In_range test
console.log("\n-------IN_RANGE TEST-------");
console.log(nodes.in range(0, 0, 4));

// X_r test
console.log("\n----------X_R TEST----------");
console.log(nodes.x r(2));

// X l test
console.log("\n----------X_L TEST----------");
console.log(nodes.x_l(2));

// Y_u test
console.log("\n----------Y_U TEST----------");
console.log(nodes.y u(2));

// Y_d test
console.log("\n----------Y_D TEST----------");
console.log(nodes.y d(5));

// In_rect test
console.log("\n-------IN_RECT TEST--------");
console.log(nodes.in_rect(-1, 2, 5, 5));
console.log(nodes.in_rect(1, 0, -5, -5));
```

Результаты выполнения тестов:

```
--------CREATE TEST--------
Точка с именем "C" уже присутствует в хранилище.

---------ХРАНИЛИЩЕ---------
A 0 0
B 4 4
C 0 4
D -4 -4
E 0 -4
--------------------------


---------READ TEST---------
{ name: 'A', x: 0, y: 0 }
{ name: 'D', x: -4, y: -4 }
Точка с именем "F" не найдена.
undefined

--------UPDATE TEST--------
Точка с названием "F" не найдена.
Точка на именем "B" уже присутствует в хранилище.

---------ХРАНИЛИЩЕ---------
```

```
A 0 -1
B 4 4
C 0 4
D -4 -4
E 0 -4
---------------------------


--------DELETE TEST--------
Точка с именем "F" не найдена.

---------ХРАНИЛИЩЕ---------
B 4 4
C 0 4
D -4 -4
E 0 -4
---------------------------


-----MAX_DISTANCE TEST-----
[ { name: 'B', x: 4, y: 4 }, { name: 'D', x: -4, y: -4 } ]

-------IN_RANGE TEST-------
[ { name: 'C', x: 0, y: 4 }, { name: 'E', x: 0, y: -4 } ]

---------X_R TEST----------
[ { name: 'B', x: 4, y: 4 } ]

---------X_L TEST----------
[
  { name: 'C', x: 0, y: 4 },
  { name: 'D', x: -4, y: -4 },
  { name: 'E', x: 0, y: -4 }
]

---------Y U TEST----------
[ { name: 'B', x: 4, y: 4 }, { name: 'C', x: 0, y: 4 } ]

---------Y_D TEST----------
[
  { name: 'B', x: 4, y: 4 },
  { name: 'C', x: 0, y: 4 },
  { name: 'D', x: -4, y: -4 },
  { name: 'E', x: 0, y: -4 }
]

-------IN RECT TEST--------
[ { name: 'B', x: 4, y: 4 }, { name: 'C', x: 0, y: 4 } ]
[ { name: 'D', x: -4, y: -4 }, { name: 'E', x: 0, y: -4 } ]
```

Задание 2.1

Листинг программы:

```
"use strict"

class point {
    constructor(a, b) {
        this.a = a;
        this.b = b;
    }

    output() {
        console.log('(' + this.a + ' ; ' + this.b + ')');
    }
}

class line segment {
    constructor(a1, b1, a2, b2) {
        this.from = new point(a1, b1);
        this.to = new point(a2, b2);
    }

    output() {
        console.log('(' + this.from.a + ' ; ' + this.from.b + ') ' + '(' + this.to.a + ' ; ' +
this.to.b + ')');
    }

    len() {
        let val = Math.sqrt(Math.pow(this.from.a - this.to.a, 2) + Math.pow(this.from.b -
this.to.b, 2));
        return val;
    }
}

let line = new line segment(0, 0, 5, 3);
line.output();
console.log(line.len());
```

Результаты выполнения тестов:

```
(0 ; 0) (5 ; 3)
5.830951894845301
```

Задание 2.2

Листинг программы:

```
"use strict"

class triangle {
    constructor(a, b, c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    is valid() {
    return this.a < this.b + this.c && this.b < this.a + this.c && this.c < this.a + this.b;

    }

    perimeter() {
        if (this.is_valid()) {
            return (this.a + this.b + this.c);
        }
        return undefined;
    }

    square() {
        if (this.is_valid()) {
            let p = this.perimeter() / 2;
            return (Math.sqrt(p * (p - this.a) * (p - this.b) * (p - this.c)));
        }
        return undefined;
    }

    is right() {
        if (this.is valid()) {
            let hyp = Math.max(this.a, this.b, this.c);
            let kh_min = Math.min(this.a, this.b, this.c);
            let kh_sr = this.perimeter() - hyp - kh_min;
            if (Math.pow(hyp, 2) === Math.pow(kh_min, 2) + Math.pow(kh_sr, 2)) {
                return true;
            }
        }
        return false;
    }
}

console.log("\nНесуществующий треугольник. Стороны 4, 2, 2.")
let tr = new triangle(4, 2, 2);
console.log("Существует: ", tr.is_valid());
console.log("Периметр: ", tr.perimeter());
console.log("Площадь: ", tr.square());
console.log("Прямоугольный: ", tr.is_right());

console.log("\nНепрямоугольный треугольник. Стороны 2, 2, 2.")
tr = new triangle(2, 2, 2);
console.log("Существует: ", tr.is_valid());
console.log("Периметр: ", tr.perimeter());
console.log("Площадь: ", tr.square());
console.log("Прямоугольный: ", tr.is right());

console.log("\nПрямоугольный треугольник треугольник. Стороны 6, 8, 10.")
tr = new triangle(6, 8, 10);
console.log("Существует: ", tr.is valid());
console.log("Периметр: ", tr.perimeter());
console.log("Площадь: ", tr.square());
console.log("Прямоугольный: ", tr.is_right());
```

Результаты выполнения тестов:

```
Несуществующий треугольник. Стороны 4, 2, 2.
Существует:  false
Периметр:  undefined
Площадь:  undefined
Прямоугольный:  false

Непрямоугольный треугольник. Стороны 2, 2, 2.
Существует:  true
Периметр:  6
Площадь:  1.7320508075688772
Прямоугольный:  false

Прямоугольный треугольник треугольник. Стороны 6, 8, 10.
Существует:  true
Периметр:  24
Площадь:  24
Прямоугольный:  true
```

Задание 2.3

Листинг программы:

```
"use strict"

let x = 0;
let delay = 2000;

setTimeout(function tick() {
    x++;
    console.log(x);
    if (x === 20) {
        x = 0;
        delay = 2000;
    }
    if (x === 10) {
        delay = 1000;
    }
    setTimeout(tick, delay);
  }, 0);
```

Программа выводит на экран значения от 1 до 20 с разными задержками, показать это в отчёте нельзя.

Вывод:

В ходе данной лабораторной работы я установил NodeJS, научился писать программы и их запускать. В заданиях 1.X я использовал объекты для выполнения поставленной задачи. В заданиях 2.1 и 2.2 я научился использовать классы в JavaScript и в задании 2.3 познакомился с контролем выполнения программы.