|  |  |
| --- | --- |
| 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 **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 1 |

**Название:**

Базовые знания JavaScript

**Дисциплина:** Архитектура ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.Ю. Попов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

Оглавление

[Цель работы: 3](#_Toc52194168)

[Task 1 4](#_Toc52194169)

[Задание 1 4](#_Toc52194170)

[Задание 2 10](#_Toc52194172)

[Задание 3 15](#_Toc52194174)

[Task 2 21](#_Toc52194176)

[Задание 1 21](#_Toc52194177)

[Задание 2 23](#_Toc52194178)

[Задание 3 26](#_Toc52194179)

[Вывод: 28](#_Toc52194180)

# 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 кода

"use strict";

function check\_surname(surname, children) {

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

        if (children[i].surname === surname) {

            return false;

        }

    }

    return true;

}

function create\_child(surname, age, children) {

    if (check\_surname(surname, children)) {

        children.push({surname: surname, age: age});

    }

    else {

        console.log("ERROR. The child was not created because the surname (" + surname + ") is not unique")

    }

}

function print\_one(child) {

    console.log(child);

}

function print\_all(children) {

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

        print\_one(children[i]);

    }

}

function update\_surname(new\_surname, child) {

    child.surname = new\_surname;

}

function update\_age(new\_age, child) {

    child.age = new\_age;

}

function update\_child(new\_surname, new\_age, child) {

    update\_surname(new\_surname, child);

    update\_age(new\_age, child);

}

function delete\_child(surname, children) {

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

        if (children[i].surname === surname) {

            children.splice(i,1);

            return;

        }

    }

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function average\_age(children) {

    if (children.length === 0) {

        return 0

    }

    let ave = 0;

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

        ave += children[i].age;

    }

    return ave / children.length

}

function max\_age\_child(children) {

    let elder = children[0];

    for (let i = 1; i < children.length; i++) {

        if (children[i].age > elder.age)

            elder = children[i];

    }

    return elder

}

function find\_children\_by\_age(children, start, stop) {

    let found = [];

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

        if (children[i].age >= start && children[i].age <= stop) {

            found.push({surname:children[i].surname, age: children[i].age});

        }

    }

    return found;

}

function find\_children\_by\_first\_symbol(children, s) {

    let found = [];

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

        if (children[i].surname[0] === s)

            found.push({surname:children[i].surname, age: children[i].age});

    }

    return found;

}

function find\_children\_by\_min\_len(children, min\_len) {

    let found = [];

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

        if (children[i].surname.length > min\_len)

            found.push({surname:children[i].surname, age: children[i].age});

    }

    return found;

}

function find\_children\_by\_vowel(children) {

    let vowel = ['A', 'E', 'U', 'Y', 'I', 'O'];

    let found = [];

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

        if (vowel.indexOf(children[i].surname[0], 0) != -1)

            found.push({surname:children[i].surname, age: children[i].age});

    }

    return found;

}

let children = [];

console.log("Creating");

create\_child("Austin", 3, children);

create\_child("Bush", 4, children);

create\_child("Conors", 5, children);

create\_child("Dyson", 8, children);

create\_child("Donaldson", 6, children);

create\_child("Ford", 7, children);

create\_child("Goodman", 6, children);

create\_child("Harrison", 3, children);

create\_child("Mackenzie", 4, children);

console.log("\nAll children:");

print\_all(children);

console.log("\nChild update");

update\_child("Adamson", 2, children[0]);

print\_one(children[0]);

console.log("\nDeleting Mackenzie");

delete\_child("Mackenzie", children);

console.log("\nAll children:");

print\_all(children);

console.log("\nAverange age:")

console.log(average\_age(children));

console.log("\nOlder child:");

console.log(max\_age\_child(children));

console.log("\nNeed children 1-4:");

console.log(find\_children\_by\_age(children, 1, 4));

console.log("\nNeed children 5-10:");

console.log(find\_children\_by\_age(children, 5, 10));

console.log("\nChildren Begin Surname D:");

console.log(find\_children\_by\_first\_symbol(children, 'D'));

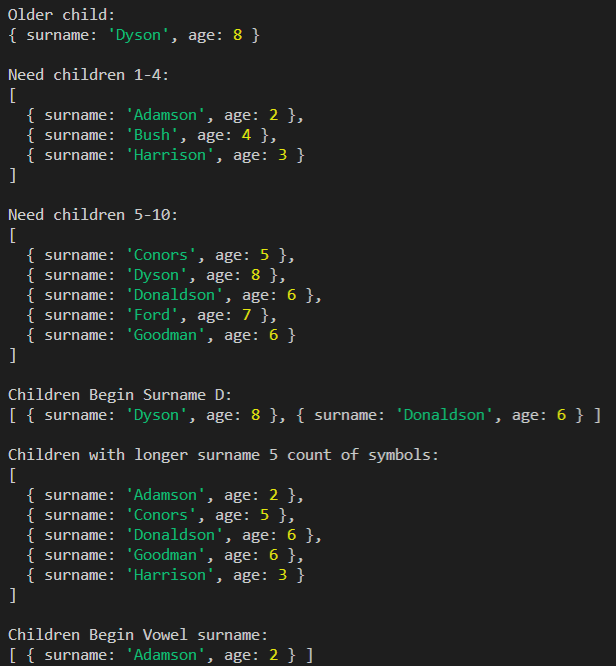
console.log("\nChildren with longer surname 5 count of symbols:");

console.log(find\_children\_by\_min\_len(children, 5));

console.log("\nChildren Begin Vowel surname:");

console.log(find\_children\_by\_vowel(children));





### 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";

function check\_card(card, student) {

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

        if (student[i].card === card) {

            console.log(student[i]);

            return false;

        }

    }

    return true;

}

// создание студента

function create\_student(group, card, mark, student) {

    if (check\_card(card, student)) {

        student.push( {

            group: group,

            card: card,

            mark: mark

        });

    }

    else {

        console.log("ERROR. The student was not created because the card (" + card + ") is not unique")

    }

}

// прочтение информации о студенте

function print\_one(student) {

    console.log(student);

}

// прочтение информации о студентах

function print\_all(student) {

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

        print\_one(student[i]);

    }

}

// обновить группу студента

function update\_group(newGroup, student) {

    student.group = newGroup;

}

// обновить студенческий билет студента

function update\_card(card, student) {

    student.card = card;

}

// обновить оценки студента

function update\_mark(newMark, student) {

    student.mark = newMark;

}

// обновить данные студента

function update\_all(new\_group, new\_card, new\_mark, student) {

    update\_group(new\_group, student);

    update\_card(new\_card, student);

    update\_mark(new\_mark, student);

}

// удалить студента

function delete\_student(card, student) {

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

        if (student[i].card === card) {

            student.splice(i,1);

            return;

        }

    }

}

// нвхождение средней оценки студента

function averange\_mark(student) {

    if (student.mark.length == 0)

        return 0;

    let mark = 0;

    for (let i = 0; i < student.mark.length; i++) {

        mark += student.mark[i];

    }

    return mark / student.mark.length

}

// получение информации о студентах в заданной группе

function get\_student(group, student) {

    let groups = [];

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

        if (student[i].group === group) {

            groups.push(student[i]);

        }

    }

    return groups;

}

// максимальное кол-во оценок у студентов

function find\_max\_cnt\_mark(student) {

    let mcount = 0;

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

        if (student[i].mark.length > mcount) {

            mcount = student[i].mark.length;

        }

    }

    return mcount;

}

// получение студента, у которого наибольшее количество оценок в заданной группе

function find\_student\_with\_max\_count(student) {

    let students = [];

    let mcount = find\_max\_cnt\_mark(student);

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

        if (student[i].mark.length == mcount) {

            students.push(student[i]);

        }

    }

    return students;

}

// получение студента, у которого нет оценок

function find\_student\_without\_mark(student) {

    let students = [];

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

        if (student[i].mark.length == 0) {

            students.push(student[i]);

        }

    }

    return students;

}

let student = [];

console.log("Creating");

create\_student("IU7-52", 10, [5, 4, 3], student);

create\_student("IU7-51", 11, [5, 3], student);

create\_student("IU7-52", 12, [5], student);

create\_student("IU7-53", 13, [5, 2, 5], student);

create\_student("IU7-53", 14, [4, 3, 3], student);

create\_student("IU7-52", 15, [4, 3], student);

create\_student("IU7-51", 16, [2, 1], student);

create\_student("IU7-53", 17, [2, 2], student);

create\_student("IU7-54", 18, [], student);

print\_all(student);

console.log("\nDelete Student 12");

delete\_student(12, student);

print\_all(student);

console.log("\nUpdate Student[5]");

console.log("Before");

print\_all(student[5]);

update\_all("IU7-54", 118, [3, 4, 5], student[5]);

console.log("After");

print\_one(student[5]);

console.log("\nAverange mark student[0]");

console.log(averange\_mark(student[0]));

console.log("\nAverange mark student[6]");

console.log(averange\_mark(student[6]));

console.log("\nStudents from group IU7-52");

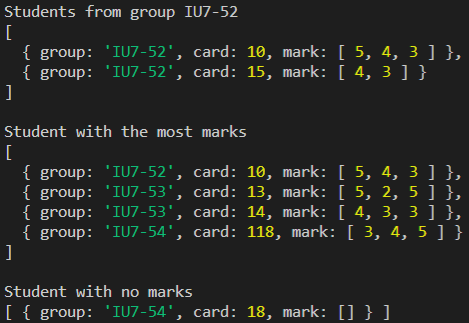
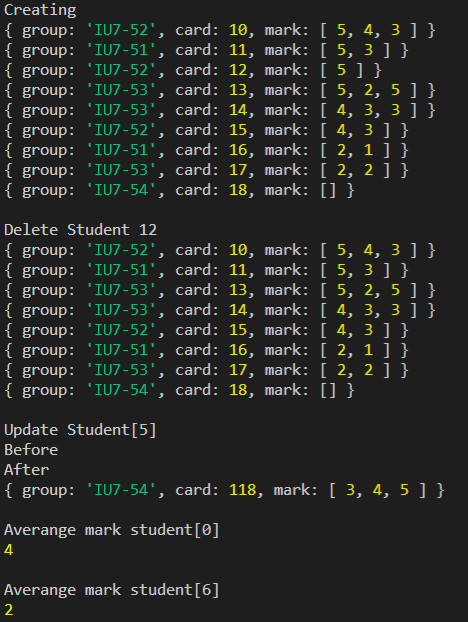
console.log(get\_student("IU7-52", student));

console.log("\nStudent with the most marks");

console.log(find\_student\_with\_max\_count(student));

console.log("\nStudent with no marks");

console.log(find\_student\_without\_mark(student));



### Задание 3

**Условие задачи:**

Создать хранилище в оперативной памяти для хранения точек.

Необходимо 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";

// проверка уникальности имени точки

function check\_name(name, points) {

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

        if (points[i].name === name) {

            return false;

        }

    }

    return true;

}

// создание точки

function create\_point(name, pointX, pointY, points) {

    if (check\_name(name, points)) {

        points.push( {

            name: name,

            x: pointX,

            y: pointY

        });

    }

    else {

        console.log("ERROR. The Point was not created because the name (" + name + ") is not unique")

    }

}

// чтение информации точки

function print\_one(point) {

    console.log(point);

}

// чтение информации точек

function print\_all(points) {

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

        print\_one(points[i]);

    }

}

// обновить имя точки

function update\_name(new\_name, point) {

    point.name = new\_name;

}

// обновить абсциссу точки

function update\_x(new\_x, point) {

    point.x = new\_x;

}

// обновить ординату точки

function update\_y(new\_y, point) {

    point.y = new\_y;

}

// обновить все данные о точки

function update\_all(new\_name, new\_x, new\_y, point) {

    update\_name(new\_name, point);

    update\_x(new\_x, point);

    update\_y(new\_y, point);

}

// удалить точку

function delete\_point(name, points) {

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

        if (points[i].name === name) {

            points.splice(i,1);

            return;

        }

    }

}

// расстояние между двумя точками

function distance(point\_a, point\_b) {

    let dx = point\_b.x - point\_a.x;

    let dy = point\_b.y - point\_a.y;

    return Math.sqrt(dx \* dx + dy \* dy);

}

// получение двух точек, между которыми наибольшее расстояние

function find\_max\_distance(points) {

    let mdist = 0;

    let find\_point = [];

    let dist;

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

        for (let j = i + 1; j < points.length; j++) {

            dist = distance(points[i], points[j]);

            if (dist > mdist) {

                find\_point = [];

                find\_point.push([points[i],points[j]]);

                mdist = dist;

            } else if (dist == mdist) {

                find\_point.push([points[i],points[j]]);

            }

        }

    }

    return find\_point;

}

// получение точек, находящихся от заданной точки на расстоянии, не превышающем заданную константу

function find\_points\_less\_dist(set\_point, dist, points) {

    let find\_point = [];

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

        if (points[i] != set\_point) {

            if (distance(points[i], set\_point) <= dist) {

                find\_point.push(points[i]);

            }

        }

    }

    return find\_point;

}

// получение точек, находящихся выше / ниже оси абсцисс

function find\_point\_ox(dir, points) {

    let find\_point = [];

    if (dir === "above" || dir === "below") {

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

            if (dir === "above") {

                if (points[i].y > 0) {

                    find\_point.push(points[i]);

                }

            } else {

                if (points[i].y < 0) {

                    find\_point.push(points[i]);

                }

            }

        }

    } else {

        console.log("incorrect direction");

    }

    return find\_point;

}

// получение точек, находящихся левее / правее оси ординат

function find\_point\_oy(dir, points) {

    let find\_point = [];

    if (dir === "left" || dir === "right") {

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

            if (dir === "left") {

                if (points[i].x < 0) {

                    find\_point.push(points[i]);

                }

            } else {

                if (points[i].x > 0) {

                    find\_point.push(points[i]);

                }

            }

        }

    } else {

        console.log("incorrect direction");

    }

    return find\_point;

}

// получение точек, входящих внутри заданной прямоугольной зоны

// прямоугольник задается точкой нижнего левого угла, шириной и высотой

function rect\_check(x, y, width, height, points) {

    let find\_point = [];

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

        if (points[i].x > x && points[i].x < (x + width) && points[i].y > y && points[i].y < (y + height)) {

            find\_point.push(points[i]);

        }

    }

    return find\_point;

}

let points = [];

console.log("Creating");

create\_point("1", -1, 1, points);

create\_point("2", 1, 1, points);

create\_point("3", 1, -1, points);

create\_point("2", -1, -1, points);

create\_point("4", -10, -10, points);

create\_point("5", 0, 0, points);

create\_point("6", 0, 0, points)

print\_all(points);

console.log("\nDeleting five point");

delete\_point("5", points);

print\_all(points);

console.log("\nUpdate our point");

print\_one(points[3]);

update\_all("4", -1, -1, points[3]);

print\_one(points[3]);

console.log("\nFind point with max distance");

console.log(find\_max\_distance(points));

console.log("\nFind point with set distance");

console.log("set distanse 5:");

console.log(find\_points\_less\_dist(points[3], 5, points));

console.log("set distanse 2:");

console.log(find\_points\_less\_dist(points[3], 2, points));

console.log("\nFind point above from axis OX");

console.log(find\_point\_ox("above", points));

console.log("\nFind point below from axis OX");

console.log(find\_point\_ox("below", points));

console.log("\nFind point left from axis OY");

console.log(find\_point\_oy("left", points));

console.log("\nFind point right from axis OY");

console.log(find\_point\_oy("right", points));

console.log("\nCheck if incorrect direction")

console.log(find\_point\_ox("1", points));

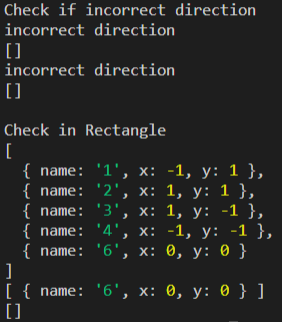
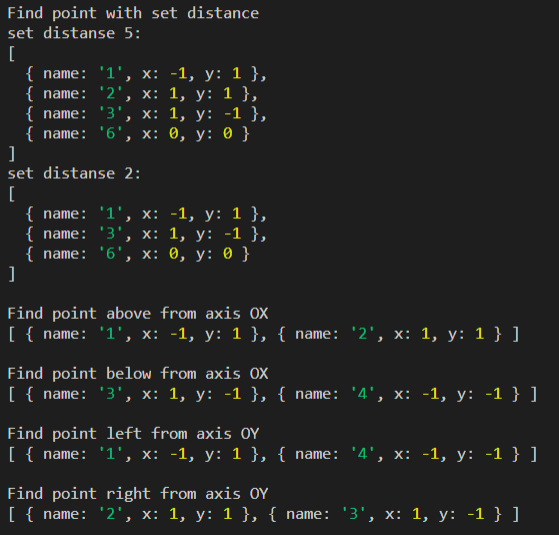
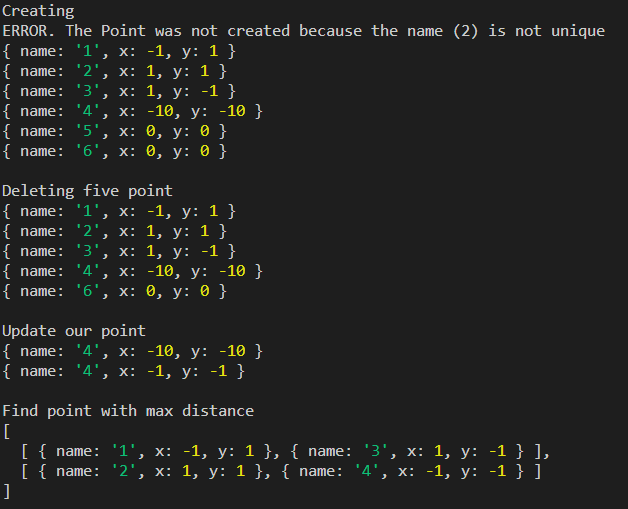
console.log(find\_point\_oy("2", points));

console.log("\nCheck in Rectangle");

console.log(rect\_check(-2, -2, 5, 5, points));

console.log(rect\_check(-1, -1, 2, 2, points));

console.log(rect\_check(-0.5, -0.5, 0.1, 0.1, points));



## Task 2

### Задание 1

**Условие задания:**

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.

**Листинг кода:**

"use strict";

class Point {

    constructor(x,  y) {

        this.x = x;

        this.y = y;

    }

    console\_log() {

        let start = " point coord: \n"

        let ox = "   x: " + this.x + "\n";

        let oy = "   y: " + this.y + "\n";

        let message = start + ox + oy;

        console.log(message);

    }

}

class Cut {

    constructor(x1, y1, x2, y2) {

        this.start = new Point(x1, y1);

        this.end = new Point(x2, y2);

    }

    console\_log() {

        console.log("First point: ");

        this.start.console\_log();

        console.log("Second point: ");

        this.end.console\_log();

    }

    get\_length() {

        let dx = this.end.x - this.start.x;

        let dy = this.end.y - this.start.y;

        return Math.sqrt(dx \* dx + dy \* dy);

    }

}

console.log("Start program.");

let point = new Point(3.5, 4);

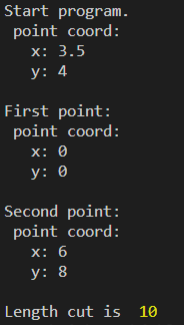
point.console\_log();

let cut = new Cut (0, 0, 6, 8);

cut.console\_log();

let len = cut.get\_length();

console.log("Length cut is ", len);



### 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;

    }

    console\_log() {

        let start = " triangle len: \n"

        let a = "   a: " + this.a + "\n";

        let b = "   b: " + this.b + "\n";

        let c = "   b: " + this.c + "\n";

        let message = start + a + b + c;

        console.log(message);

    }

    triangle\_check() {

        if ((this.a + this.b > this.c) && (this.a + this.c > this.b) && (this.c + this.b > this.a)) {

            return true;

        }

        return false;

    }

    hypotenuse(leg1, leg2, hypotense) {

        if ((leg1 \* leg1 + leg2 \* leg2) == hypotense \* hypotense) {

            return true;

        }

        return false;

    }

    rectangular() {

        if (this.triangle\_check()) {

            let tmp = Math.max(this.a, this.b, this.c);

            if (tmp == this.a) {

                return this.hypotenuse(this.b, this.c, tmp);

            } else if (tmp == this.b) {

                return this.hypotenuse(this.a, this.c, tmp);

            } else {

                return this.hypotenuse(this.a, this.b, tmp);

            }

        }

        return NaN;

    }

    perimetr() {

        if (this.triangle\_check()) {

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

        }

        return NaN;

    }

    square() {

        if (this.triangle\_check()) {

            let p = this.perimetr() / 2;

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

        }

        return NaN;

    }

}

let triangle = new Triangle(10, 9, 11);

console.log("Data: 10, 9, 11");

console.log("Is it triangle?");

console.log(triangle.triangle\_check());

console.log("Perimetr = ", triangle.perimetr());

console.log("Square = ", triangle.square());

console.log("Is it rectangular triangle?");

console.log(triangle.rectangular());

let triangle1 = new Triangle(15, 15, 15);

console.log("Data: 15, 15, 15");

console.log("Is it triangle?");

console.log(triangle1.triangle\_check());

console.log("Perimetr = ", triangle1.perimetr());

console.log("Square = ", triangle1.square());

console.log("Is it rectangular triangle?");

console.log(triangle1.rectangular());

let triangle2 = new Triangle(3, 4, 5);

console.log("Data: 3, 4, 5");

console.log("Is it triangle?");

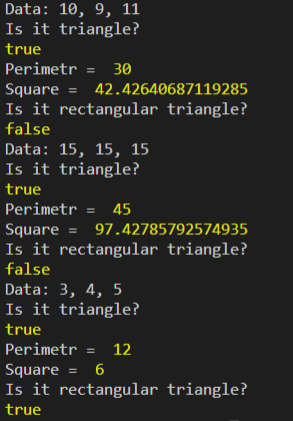
console.log(triangle2.triangle\_check());

console.log("Perimetr = ", triangle2.perimetr());

console.log("Square = ", triangle2.square());

console.log("Is it rectangular triangle?");

console.log(triangle2.rectangular());



### 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 count = 0;

let needCount = 2;

function printNumber(delay, stage){

    let num = 0;

    let interval = setInterval(() => {

        if (num >= 0) {

            num++;

            if (stage === 0) {

                console.log(num);

            } else {

                console.log(num + 10);

            }

        }

        if (num === 10) {

            clearInterval(interval);

            num = 0;

            if (stage === 0) {

                printNumber(1000, 1);

            } else {

                count++;

                if (count != needCount) {

                    printNumber(2000, 0);

                }

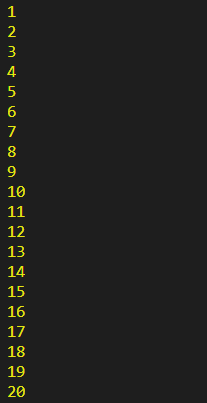
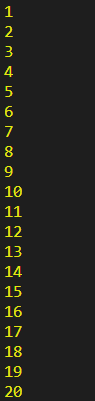
            }

        };

    }, delay);

}

printNumber(2000, 0);

****

## Вывод:

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