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**

О Т Ч Е Т

По лабораторной работе № 1

Название: Базовые знания JavaScript

Дисциплина: Архитектура ЭВМ

Студент ИУ7-52Б (Группа) _____ (Подпись, дата) Н.А. Гарасев (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) А.Ю. Попов (И.О. Фамилия)

Москва, 2020

Оглавление

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 (" + su
rname + ") 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));
```

```
Creating

All children:
{ surname: 'Austin', age: 3 }
{ surname: 'Bush', age: 4 }
{ surname: 'Conors', age: 5 }
{ surname: 'Dyson', age: 8 }
{ surname: 'Donaldson', age: 6 }
{ surname: 'Ford', age: 7 }
{ surname: 'Goodman', age: 6 }
{ surname: 'Harrison', age: 3 }
{ surname: 'Mackenzie', age: 4 }

Child update
{ surname: 'Adamson', age: 2 }

Deleting Mackenzie

All children:
{ surname: 'Adamson', age: 2 }
{ surname: 'Bush', age: 4 }
{ surname: 'Conors', age: 5 }
{ surname: 'Dyson', age: 8 }
{ surname: 'Donaldson', age: 6 }
{ surname: 'Ford', age: 7 }
{ surname: 'Goodman', age: 6 }
{ surname: 'Harrison', age: 3 }

Averange age:
5.125
```

```
Older child:
{ surname: 'Dyson', age: 8 }

Need children 1-4:
[
  { surname: 'Adamson', age: 2 },
  { surname: 'Bush', age: 4 },
  { surname: 'Harrison', age: 3 }
]

Need children 5-10:
[
  { surname: 'Conors', age: 5 },
  { surname: 'Dyson', age: 8 },
  { surname: 'Donaldson', age: 6 },
  { surname: 'Ford', age: 7 },
  { surname: 'Goodman', age: 6 }
]

Children Begin Surname D:
[ { surname: 'Dyson', age: 8 }, { surname: 'Donaldson', age: 6 } ]

Children with longer surname 5 count of symbols:
[
  { surname: 'Adamson', age: 2 },
  { surname: 'Conors', age: 5 },
  { surname: 'Donaldson', age: 6 },
  { surname: 'Goodman', age: 6 },
  { surname: 'Harrison', age: 3 }
]

Children Begin Vowel surname:
[ { surname: 'Adamson', age: 2 } ]
```

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 (" + car
d + ") 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));
```

```
Creating
{ group: 'IU7-52', card: 10, mark: [ 5, 4, 3 ] }
{ group: 'IU7-51', card: 11, mark: [ 5, 3 ] }
{ group: 'IU7-52', card: 12, mark: [ 5 ] }
{ group: 'IU7-53', card: 13, mark: [ 5, 2, 5 ] }
{ group: 'IU7-53', card: 14, mark: [ 4, 3, 3 ] }
{ group: 'IU7-52', card: 15, mark: [ 4, 3 ] }
{ group: 'IU7-51', card: 16, mark: [ 2, 1 ] }
{ group: 'IU7-53', card: 17, mark: [ 2, 2 ] }
{ group: 'IU7-54', card: 18, mark: [] }

Delete Student 12
{ group: 'IU7-52', card: 10, mark: [ 5, 4, 3 ] }
{ group: 'IU7-51', card: 11, mark: [ 5, 3 ] }
{ group: 'IU7-53', card: 13, mark: [ 5, 2, 5 ] }
{ group: 'IU7-53', card: 14, mark: [ 4, 3, 3 ] }
{ group: 'IU7-52', card: 15, mark: [ 4, 3 ] }
{ group: 'IU7-51', card: 16, mark: [ 2, 1 ] }
{ group: 'IU7-53', card: 17, mark: [ 2, 2 ] }
{ group: 'IU7-54', card: 18, mark: [] }

Update Student[5]
Before
After
{ group: 'IU7-54', card: 118, mark: [ 3, 4, 5 ] }

Averange mark student[0]
4

Averange mark student[6]
2
```

```
Students from group IU7-52
[
  { group: 'IU7-52', card: 10, mark: [ 5, 4, 3 ] },
  { group: 'IU7-52', card: 15, mark: [ 4, 3 ] }
]

Student with the most marks
[
  { group: 'IU7-52', card: 10, mark: [ 5, 4, 3 ] },
  { group: 'IU7-53', card: 13, mark: [ 5, 2, 5 ] },
  { group: 'IU7-53', card: 14, mark: [ 4, 3, 3 ] },
  { group: 'IU7-54', card: 118, mark: [ 3, 4, 5 ] }
]

Student with no marks
[ { group: 'IU7-54', card: 18, mark: [] } ]
```

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 з
аданную 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 && po
ints[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));
```

```
Creating
ERROR. The Point was not created because the name (2) is not unique
{ name: '1', x: -1, y: 1 }
{ name: '2', x: 1, y: 1 }
{ name: '3', x: 1, y: -1 }
{ name: '4', x: -10, y: -10 }
{ name: '5', x: 0, y: 0 }
{ name: '6', x: 0, y: 0 }

Deleting five point
{ name: '1', x: -1, y: 1 }
{ name: '2', x: 1, y: 1 }
{ name: '3', x: 1, y: -1 }
{ name: '4', x: -10, y: -10 }
{ name: '6', x: 0, y: 0 }

Update our point
{ name: '4', x: -10, y: -10 }
{ name: '4', x: -1, y: -1 }

Find point with max distance
[
  [ { name: '1', x: -1, y: 1 }, { name: '3', x: 1, y: -1 } ],
  [ { name: '2', x: 1, y: 1 }, { name: '4', x: -1, y: -1 } ]
]
```

```
Find point with set distance
set distanse 5:
[
  { name: '1', x: -1, y: 1 },
  { name: '2', x: 1, y: 1 },
  { name: '3', x: 1, y: -1 },
  { name: '6', x: 0, y: 0 }
]
set distanse 2:
[
  { name: '1', x: -1, y: 1 },
  { name: '3', x: 1, y: -1 },
  { name: '6', x: 0, y: 0 }
]

Find point above from axis OX
[ { name: '1', x: -1, y: 1 }, { name: '2', x: 1, y: 1 } ]

Find point below from axis OX
[ { name: '3', x: 1, y: -1 }, { name: '4', x: -1, y: -1 } ]

Find point left from axis OY
[ { name: '1', x: -1, y: 1 }, { name: '4', x: -1, y: -1 } ]

Find point right from axis OY
[ { name: '2', x: 1, y: 1 }, { name: '3', x: 1, y: -1 } ]
```

```
Check if incorrect direction
incorrect direction
[]
incorrect direction
[]

Check in Rectangle
[
  { name: '1', x: -1, y: 1 },
  { name: '2', x: 1, y: 1 },
  { name: '3', x: 1, y: -1 },
  { name: '4', x: -1, y: -1 },
  { name: '6', x: 0, y: 0 }
]
[ { name: '6', x: 0, y: 0 } ]
[]
```

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);
```

```
Start program.
 point coord:
   x: 3.5
   y: 4

First point:
 point coord:
   x: 0
   y: 0

Second point:
 point coord:
   x: 6
   y: 8

Length cut is  10
```

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());
```

```
Data: 10, 9, 11
Is it triangle?
true
Perimetr =  30
Square =  42.42640687119285
Is it rectangular triangle?
false
Data: 15, 15, 15
Is it triangle?
true
Perimetr =  45
Square =  97.42785792574935
Is it rectangular triangle?
false
Data: 3, 4, 5
Is it triangle?
true
Perimetr =  12
Square =  6
Is it rectangular triangle?
true
```

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:

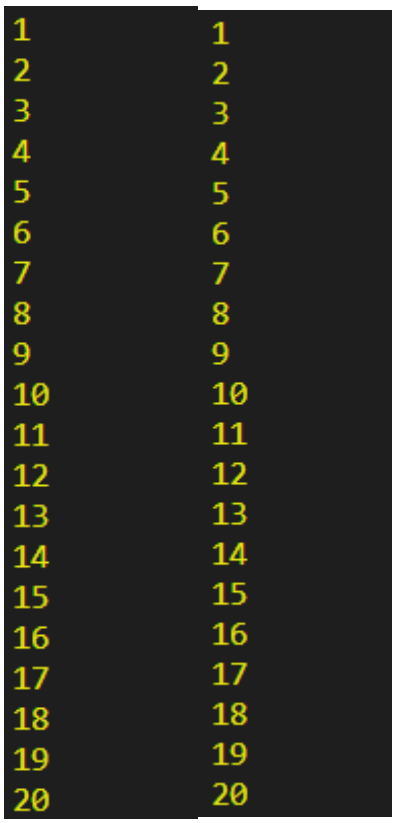
```
"use strict";

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.