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 **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 1

Дисциплина: Архитектура ЭВМ

Студент	ИУ7-52Б		В.А. Иванов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			А.Ю. Попов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2020

Цель работы

Целью лабораторной работы является ознакомление и практическое закрепление основ языка JavaScript.

Задание 1.1 Создать хранилище в оперативной памяти для хранения информации о детях.

Условие

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

Программная реализация

```
"use strict";
// Хранилище информации о детях
let data = [];

function print_kid(kid)
{
    console.log("> " + kid.surname + " : " + kid.age);
}

function is_surname_in(surname_)
{
    for (let i=0; i < data.length; i++)
    {
        if (data[i].surname == surname_)
                return true;
    }
    return false;
}
function find_index(surname_)
```

```
{
    for (let i=0; i < data.length; i++)
    {
        if (data[i].surname == surname_)
                return i;
    }
    return -1;
}


function average_age()  // Вычисление среднего возраста
{
    let sum = 0;
    for (let i=0; i < data.length; i++)
        sum += data[i].age;
    return parseInt(sum/data.length);
}
function oldest_kid()   // Поиск самого старшего ребёнка
{
    if (data.length == 0) return null;
    let max_i = 0;
    for (let i=1; i < data.length; i++)
    {
        if (data[i].age > data[max_i].age)
            max_i = i;
    }
    return data[max_i];
}
function in_adge_range(begin, end)  // Дети с возрастом в заданном отрезке
{
    let count = 0;
    console.log("Дети с возрастом от " + begin + " до " + end + " лет:");
    for (let i=0; i < data.length; i++)
    {
        if (begin <= data[i].age && data[i].age <= end)
        {
            print_kid(data[i]);
            count++;
        }
    }

    if (!count)
        console.log("Дети с заданным возрастом не найдены");
}
function surname_at(char)   // Дети с фамилией на заданную букву
{
    let count = 0;
    console.log("Дети с фамилией на букву " + char + ":");
    for (let i=0; i < data.length; i++)
    {
        if (data[i].surname[0] === char)
        {
            print_kid(data[i]);
            count++;
        }
    }

    if (!count)
        console.log("Дети с заданным параметром не найдены");
}
function surname_longer_len(len)    // Дети с фамилией длинне, чем заданного кол-ва символов
{
    let count = 0;
    console.log("Дети с фамилией длиннее " + len + " символов:");
    for (let i=0; i < data.length; i++)
    {
        if (data[i].surname.length > len)
        {
```

```
            print_kid(data[i]);
            count++;
        }
    }
    if (!count)
        console.log("Дети с заданным параметром не найдены");
}
function surname_vowel()    // Дети с фамилией на гласную букву
{
    let count = 0;
    let vowel = "ауоыиэяюёе";
    console.log("Дети с фамилией на гласную букву" + ":");
    for (let i=0; i < data.length; i++)
    {
        if (vowel.search(data[i].surname[0].toLowerCase()) != -1)
        {
            print_kid(data[i]);
            count++;
        }
    }

    if (!count)
        console.log("Дети с заданным параметром не найдены");
}

/// CDIO data
function CREATE(surname_, age_)
{
    if (!is_surname_in(surname_))
    {
        let new_child =
        {   surname : surname_,
            age : age_  };
        data.push(new_child);
    }
    else
    {
        console.log("Запись не добавлена: Фамилия \"" + surname_ + "\" уже в хранилище.");
    }
}
function DELETE(surname_)
{
    let i = find_index(surname_);
    if (i != -1)
    {
        data.splice(i, 1);
    }
    else
    {
        console.log("Запись не удалена: Фамилия \"" + surname_ + "\" не в хранилище.");
    }
}
function READ()
{
    if (!data.length) {
        console.log("Хранилище не содержит записей.");
        return;
    }

    console.log("\nСписок всех детей (> Фамилия : Возраст):");
    for (let i=0; i < data.length; i++)
    {
        print_kid(data[i]);
    }
}
function UPDATE(pre_surname, surname_, age_)
{
    let i = find_index(pre_surname);
```

```
    if (i != -1)
    {
        data[i].surname = surname_;
        data[i].age = age_;
    }
    else
    {
        console.log("Запись не обновлена: Фамилия \"" + pre_surname + "\" не в хранилище.");
    }
}
function CLEAR()
{
    data = [];
}
```

Тесты

Тест №1:

```
function test_1()
{
    console.log("\nСоздание записей");
    CREATE("Иванов", 20);
    CREATE("Помоев", 23);
    CREATE("Иванов", 22);
    CREATE("Сучёчков", 4);
    CREATE("Песков", 2);
    READ();

    console.log("\nУдаление записей");
    DELETE("Помоев");
    DELETE("Брянская");
    READ();

    console.log("\nОбновление записей");
    UPDATE("Брянская", "Иванов", 20);
    UPDATE("Сучёчков", "Сучков", 19);
    READ();
    CLEAR();
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_1_1.js

_____________________________________________________


Создание записей
Запись не добавлена: Фамилия "Иванов" уже в хранилище.

Список всех детей (> Фамилия : Возраст):
> Иванов : 20
> Помоев : 23
> Сучёчков : 4
> Песков : 2

Удаление записей
Запись не удалена: Фамилия "Брянская" не в хранилище.

Список всех детей (> Фамилия : Возраст):
> Иванов : 20
> Сучёчков : 4
> Песков : 2

Обновление записей
Запись не обновлена: Фамилия "Брянская" не в хранилище.

Список всех детей (> Фамилия : Возраст):
> Иванов : 20
> Сучков : 19
> Песков : 2

_____________________________________________________
```

Тест №2:

```
function test_2()
{
    CLEAR();
    CREATE("Иванов", 21);
    CREATE("Помоев", 21);
    CREATE("Буратинов", 20);
    CREATE("Сталин", 7);
    CREATE("Сучёчков", 3);
    CREATE("Усков", 2);
    READ();

    console.log("Средний возраст детей: " + average_age());

    console.log("\nСтарший ребёнок:");
    print_kid(oldest_kid());

    in_adge_range(3, 20);
    surname_at("С");
    surname_longer_len(6);
    surname_vowel();
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_1_1.js

_____________________________________________________________


Список всех детей (> Фамилия : Возраст):
> Иванов : 21
> Помоев : 21
> Буратинов : 20
> Сталин : 7
> Сучёчков : 3
> Усков : 2
Средний возраст детей: 12

Старший ребёнок:
> Иванов : 21
Дети с возрастом от 3 до 20 лет:
> Буратинов : 20
> Сталин : 7
> Сучёчков : 3
Дети с фамилией на букву С:
> Сталин : 7
> Сучёчков : 3
Дети с фамилией длиннее 6 символов:
> Буратинов : 20
> Сучёчков : 3
Дети с фамилией на гласную букву:
> Иванов : 21
> Усков : 2


_____________________________________________________________
```

Задание 1.2 Создать хранилище в оперативной памяти для хранения информации о студентах.

Условие

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

Программная реализация

```
"use strict";
// Хранилище информации о студентах
let data = [];
let student_t =
{   group : null,
    id :    null,
    marks : null };

function print_student(st)
{
    if (st.marks.length)
        console.log("> №"+ st.id + ", " + st.group + " : " + st.marks);
    else
        console.log("> №"+ st.id + ", " + st.group + " : без оценок");
}

function is_id_in(id_)
{
    for (let i=0; i < data.length; i++)
    {
        if (data[i].id == id_)
            return true;
    }
    return false;
}
function find_index(id_)
{
    for (let i=0; i < data.length; i++)
    {
        if (data[i].id == id_)
                return i;
    }
    return -1;
}
```

```
function average_mark(id_)  // Средняя оценка указанного студента
{
    let pos = find_index(id_);
    if (pos == -1)
    {
        console.log("Запись не найдена: № билета " + id_ + " нет в хранилище.");
        return;
    }

    let st = data[pos];
    if (st.marks.length != 0)
    {
        let sum = 0;
        for (let i=0; i < st.marks.length; i++)
            sum += st.marks[i];
        sum /= st.marks.length;
        console.log("Средняя оценка студента №" + st.id + " равна " + sum);
    }
    else
    {
        console.log("Студент №" + st.id + " не имеет оценок");
    }
}
function print_group(group_)    // Вывод студентов данной группы
{
    let count = 0;
    console.log("Информация о студентах группы " + group_ + ":");
    for (let i=0; i<data.length; i++)
    {
        if (data[i].group == group_)
        {
            print_student(data[i]);
            count++;
        }
    }

    if (!count)
    {
        console.log("Студенты заданной группы не найдены");
    }
}
function most_marks_group(group_)   // Получение студента с наибольшим количеством оценок в да
нной группе
{
    let st = null;
    console.log("Информация о студентах группы " + group_ + ":");
    for (let i=0; i<data.length; i++)
    {
        if (data[i].group == group_)
        {
            if (!st || st.marks.length < data[i].marks.length)
                st = data[i]
        }
    }

    if (st)
    {
        console.log("Студент группы " + group_ + " с наибольшим количеством оценок:");
        print_student(st);
    }
    else
    {
        console.log("Студенты заданной группы не найдены");
    }
}
function no_marks()     // Получение студентов без оценок
```

```
{
    let count = 0;
    console.log("Информация о студентах без оценок:");
    for (let i=0; i<data.length; i++)
    {
        if (!data[i].marks.length)
        {
            print_student(data[i]);
            count++;
        }
    }

    if (!count)
    {
        console.log("Студенты без оценок не найдены");
    }
}

/// CDIO data
function CREATE(group_, id_, marks_=[])
{
    if (!is_id_in(id_))
    {
        let new_st =
        {   group : group_,
            id :    id_,
            marks : marks_ };

        data.push(new_st);
    }
    else
    {
        console.log("Запись не добавлена: № билета " + id_ + " уже в хранилище.");
    }
}
function DELETE(id_)
{
    let i = find_index(id_);
    if (i != -1)
    {
        data.splice(i, 1);
    }
    else
    {
        console.log("Запись не удалена: № билета " + id_ + " не в хранилище.");
    }
}
function READ()
{
    if (!data.length) {
        console.log("Хранилище не содержит записей.");
        return;
    }

    console.log("\nСписок всех студентов (> №билета, группа : оценки):");
    for (let i=0; i < data.length; i++)
    {
        print_student(data[i]);
    }
}
function UPDATE(pre_id, group_, id_, marks_)
{
    let i = find_index(pre_id);
    if (i != -1)
    {
        data[i].group = group_;
        data[i].id = id_;
        data[i].marks = marks_;
```

```
    }
    else
    {
        console.log("Запись не обновлена: № билета " + pre_id + " не в хранилище.");
    }
}
function CLEAR()
{
    data = [];
}
```

Тесты:

Тест №1:

```
function test_1()
{
    CLEAR();
    console.log("\nСоздание записей");
    CREATE("IU7-21", 1, [4]);
    CREATE("IU7-11", 0, []);
    CREATE("SM21-41", 151, [5, 2, 3]);
    CREATE("BMT3-61", 151, [5, 2, 3]);
    CREATE("IU7-45", 110, [2, 2, 2]);
    CREATE("IU7-45", 210, [5, 5, 5]);
    READ();

    console.log("\Удаление записей");
    DELETE(51);
    DELETE(151);
    READ();

    console.log("\Обновление записей");
    UPDATE(2, "Nope-21", 2, []);
    UPDATE(110, "IU7-52", 110, [2, 2, 2, 3]);
    READ();
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_1_2.js

_____________________________________________________________


Создание записей
Запись не добавлена: № билета 151 уже в хранилище.

Список всех студентов (> № билета, группа : оценки):
> №1, IU7-21 : 4
> №0, IU7-11 : без оценок
> №151, SM21-41 : 5,2,3
> №110, IU7-45 : 2,2,2
> №210, IU7-45 : 5,5,5
Удаление записей
Запись не удалена: № билета 51 не в хранилище.

Список всех студентов (> № билета, группа : оценки):
> №1, IU7-21 : 4
> №0, IU7-11 : без оценок
> №110, IU7-45 : 2,2,2
> №210, IU7-45 : 5,5,5
Обновление записей
Запись не обновлена: № билета 2 не в хранилище.

Список всех студентов (> № билета, группа : оценки):
> №1, IU7-21 : 4
> №0, IU7-11 : без оценок
> №110, IU7-52 : 2,2,2,3
> №210, IU7-45 : 5,5,5
```

Тест №2:

```
function test_2()
{
    CLEAR();
    CREATE("IU7-21", 1, [4]);
    CREATE("IU7-11", 0, []);
    CREATE("SM21-41", 151, [5, 2, 3]);
    CREATE("IU7-45", 110, [2, 2, 2, 3]);
    CREATE("IU7-45", 210, [5, 5, 5]);
    CREATE("IU7-45", 220);
    READ();

    console.log("\nСредние оценки:\n");
    average_mark(1);
    average_mark(0);
    average_mark(100);
    average_mark(151);

    console.log("\nПоиск по группе:");
    print_group("IU7-45");
    print_group("IU7-42");
    print_group("SM21-41");

    console.log("\nСуденты по группы с наибольшим числом оценок:");
    most_marks_group("IU7-45");
    most_marks_group("IU7-42");
    most_marks_group("IU7-11");

    no_marks();
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_1_2.js

_____________________________________________________________


Список всех студентов (> № билета, группа : оценки):
> №1, IU7-21 : 4
> №0, IU7-11 : без оценок
> №151, SM21-41 : 5,2,3
> №110, IU7-45 : 2,2,2,3
> №210, IU7-45 : 5,5,5
> №220, IU7-45 : без оценок
> №151, SM21-41 : 5,2,3
> №110, IU7-45 : 2,2,2,3
> №210, IU7-45 : 5,5,5
> №220, IU7-45 : без оценок

Средние оценки:

Средняя оценка студента №1 равна 4
Студент №0 не имеет оценок
Запись не найдена: № билета 100 нет в хранилище.
Средняя оценка студента №151 равна 3.3333333333333335

Поиск по группе:
Информация о студентах группы IU7-45:
> №110, IU7-45 : 2,2,2,3
> №210, IU7-45 : 5,5,5
> №220, IU7-45 : без оценок
Информация о студентах группы IU7-42:
Студенты заданной группы не найдены
Информация о студентах группы SM21-41:
> №151, SM21-41 : 5,2,3
```

```
Суденты по группы с наибольшим числом оценок:
Информация о студентах группы IU7-45:
Студент группы IU7-45 с наибольшим количеством оценок:
> №110, IU7-45 : 2,2,2,3
Информация о студентах группы IU7-42:
Студенты заданной группы не найдены
Информация о студентах группы IU7-11:
Студент группы IU7-11 с наибольшим количеством оценок:
> №0, IU7-11 : без оценок
Информация о студентах без оценок:
> №0, IU7-11 : без оценок
> №220, IU7-45 : без оценок


_____________________________________________________________
```

Задание 1.3 Создать хранилище в оперативной памяти для хранения точек.

Условие

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

Программная реализация

```
"use strict";
let EPS = 1e-5;
// Хранилище информации о точек
let data = [];
let point_t =
{   x: 0,
    y: 0 };

function new_point(x_, y_)
{
    return { x: x_,     y: y_ };
}
function point_str(p)
{
    return ("(" + p.x + ", " + p.y + ")");
}
function print_point(p)
{
    console.log("> " + point_str(p)+ ";");
}


function find_index(x_, y_)
{
    for (let i=0; i<data.length; i++)
    {
        if (Math.abs(data[i].x - x_) < EPS &&
            Math.abs(data[i].y - y_) < EPS)
            return i;
    }
    return -1;
}
function find_distance(p1, p2)
{
```

```
    return Math.sqrt((p1.x - p2.x)**2 + (p1.y - p2.y)**2);
}
function max_distance()
{
    if (data.length < 2)
    {
        console.log("Недостаточно точек для операции");
        return;
    }

    let p1=0, p2=1;
    let max_d = find_distance(data[p1], data[p2]);
    for (let i=0; i<data.length; i++)
        for (let j=1+i; j<data.length; j++)
        {
            let d = find_distance(data[i], data[j]);
            if (d > max_d)
            {
                max_d = d;
                p1 = i;
                p2 = j;
            }
        }

    console.log("Максимальное расстояние: от " + point_str(data[p1]) +
    " до " + point_str(data[p2]) + " равно " + max_d);
}


function lower_ox()     // Вывод точек ниже оси OX
{
    function cmp(p) {
        return p.y < 0;
    }
    print_by_key(cmp, "Ниже оси OX");
}
function higher_ox()    // Вывод точек выше оси OX
{
    function cmp(p) {
        return p.y > 0;
    }
    print_by_key(cmp, "Выше оси OX");
}
function to_left_oy()   // Вывод точек левее оси OY
{
    function cmp(p) {
        return p.x < 0;
    }
    print_by_key(cmp, "Левее оси OY");
}
function to_right_oy()  // Вывод точек правее оси OY
{
    function cmp(p) {
        return p.x > 0;
    }
    print_by_key(cmp, "Правее оси OY");
}
function closer_then(x_, y_, dist)      // Вывод точек удалённых от (x,y) меньше, чем на dist
{
    if (dist < 0) {
        console.log("Ошибка: отрицательное расстояние");
        return;
    }
    let p0 = new_point(x_, y_);
    function cmp(p) {
        return (find_distance(p, p0) < dist)
    }
```

```
    print_by_key(cmp, "Расстояние от ("+x_+", "+y_+") не превышает " + dist);
}
function inside_zone(x_min, x_max, y_min, y_max)    // Вывод точек внутри зоны от (x_min, y_mi
n) до (x_max, y_max)
{
    if (x_min > x_max || y_min > y_max) {
        console.log("Некорректные параметры зоны");
    }
    function cmp(p){
        return (x_min<=p.x && x_max>=p.x) && (y_min<=p.y && y_max>=p.y);
    }

    print_by_key(cmp, "Внутри зоны от ("+x_min+", "+y_min+") до ("+x_max+", "+y_max+")");
}


function print_by_key(func, f_name="Признак не указан")     // Функция, выводящая точки по фун
кции-компаратору
{
    let count = 0;
    console.log("\nТочки по признаку: " + f_name +  ":");
    for (let i=0; i<data.length; i++)
    {
        let p = data[i];
        if (func(p))
        {
            print_point(p);
            count++;
        }
    }

    if (!count)
    {
        console.log("Точки не найдены");
    }
}

// CDIO data
function CREATE(x_, y_)
{
    let new_p =
    {   x: x_, y: y_};
    data.push(new_p);
}
function DELETE(x_, y_)
{
    let i = find_index(x_, y_);
    if (i != -1)
        data.splice(i, 1);
    else
        console.log("Точка (" + x_ + ", " + y_ + ") не найдена");
}
function READ()
{
    if (!data.length) {
        console.log("Хранилище не содержит записей.");
        return;
    }

    console.log("\nСписок всех точек:");
    for (let i=0; i < data.length; i++)
    {
        print_point(data[i]);
    }
}
function UPDATE(old_x, old_y, x_, y_)
{
    let i = find_index(old_x, old_y);
```

```
    if (i != -1)
    {
        data[i].x = x_;
        data[i].y = y_;
    }
    else
        console.log("Точка (" + old_x + ", " + old_y + ") не найдена");
}
function CLEAR()
{
    data = [];
}
```

Тесты

Тест №1:

```
function test_1()
{
    console.log("\nСоздание записей");
    CREATE(1, 0);
    CREATE(1, 0);
    CREATE(2, 0);
    CREATE(2, -3.14);
    CREATE(4, 0);
    READ();

    console.log("\nУдаление записей");
    DELETE(1, 0);
    DELETE(2, 0);
    DELETE(3, 0);
    READ();

    console.log("\Обновление записей");
    UPDATE(5, 5, 10, 10);
    UPDATE(4, 0, 5, 5);
    READ();
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_1_3.js

_____________________________________________________


Создание записей

Список всех точек:
> (1, 0);
> (1, 0);
> (2, 0);
> (2, -3.14);
> (4, 0);

Удаление записей
Точка (3, 0) не найдена

Список всех точек:
> (1, 0);
> (2, -3.14);
> (4, 0);
Обновление записей
Точка (5, 5) не найдена

Список всех точек:
> (1, 0);
> (2, -3.14);
> (5, 5);

_____________________________________________________
```

Тест №2:

```
function test_2()
{
    CREATE(1, 0);
    CREATE(1, 0);
    CREATE(2, 0);
    CREATE(2, -3.14);
    CREATE(4, 0);
    CREATE(-3, 5);
    CREATE(0, 0);
    READ();

    max_distance();
    closer_then(0, 0, 3);
    inside_zone(-1, 1, 0, 2);

    lower_ox();
    higher_ox();
    to_left_oy();
    to_right_oy();
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_1_3.js

_____________________________________________________________

Список всех точек:
> (1, 0);
> (1, 0);
> (2, 0);
> (2, -3.14);
> (4, 0);
> (-3, 5);
> (0, 0);
Максимальное расстояние: от (2, -3.14) до (-3, 5) равно 9.552989061021687

Точки по признаку: Расстояние от (0, 0) не превышает 3:
> (1, 0);
> (1, 0);
> (2, 0);
> (0, 0);

Точки по признаку: Внутри зоны от (-1, 0) до (1, 2):
> (1, 0);
> (1, 0);
> (0, 0);
```

```
Точки по признаку: Ниже оси OX:
> (2, -3.14);

Точки по признаку: Выше оси OX:
> (-3, 5);

Точки по признаку: Левее оси OY:
> (-3, 5);

Точки по признаку: Правее оси OY:
> (1, 0);
> (1, 0);
> (2, 0);
> (2, -3.14);
> (4, 0);
_____________________________________________________________
```

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 также метод получения длины отрезка.

Программная реализация

```
class Point {
    constructor(x, y, z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    printFields() {
        console.log(`(${this.x}, ${this.y}, ${this.z})`);
    }
}

class Segment {
    constructor(p1, p2) {
        this.p1 = p1;
        this.p2 = p2;
    }

    getLength() {
        return Math.sqrt(
            (this.p1.x - this.p2.x)**2 +
            (this.p1.y - this.p2.y)**2 +
            (this.p1.z - this.p2.z)**2
        );
    }

    printFields() {
        console.log("Точки отрезка:");
        this.p1.printFields();
        this.p2.printFields();
    }
}
```

Тесты

Тест №1:

```
function main() {
    let p1 = new Point(0, 0, 0);
    let p2 = new Point(1, 0, 2);
    let p3 = new Point(-1.141, 2, 3.1);
    let p4 = new Point(0, 0, 0);

    console.log("Точки:");
    p1.printFields();
    p2.printFields();
    p3.printFields();
    p4.printFields();

    console.log("\nОтрезок №1:");
    let s = new Segment(p3, p2);
    s.printFields();
    console.log("Длина отрезка = " + s.getLength());

    console.log("\nОтрезок №2:");
    s = new Segment(p1, p4);
    s.printFields();
    console.log("Длина отрезка = " + s.getLength());
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node task_2_1
Точки:
(0, 0, 0)
(1, 0, 2)
(-1.141, 2, 3.1)
(0, 0, 0)

Отрезок №1:
Точки отрезка:
(-1.141, 2, 3.1)
(1, 0, 2)
Длина отрезка = 3.1295176944698686

Отрезок №2:
Точки отрезка:
(0, 0, 0)
(0, 0, 0)
Длина отрезка = 0
```

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

Программная реализация

```
function isEqual(a, b) {
    return Math.abs(a-b) < Number.EPSILON;
}

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

    isExist() {    // Проверка возможности существования треугольника
        if (this.a>0 && this.b>0 && this.c>0) {
            let m = Math.max(this.a, this.b, this.c);
            return 2*m < this.getPerimeter();
        }
        else {
            return false;
        }
    }
    isRight() {     // Проверка прямоугольности
        let hypotenuse_2 = Math.max(this.a, this.b, this.c) ** 2;
        let cathetus_sum2 = this.a**2 + this.b**2 + this.c**2 - hypotenuse_2;
        return isEqual(cathetus_sum2, hypotenuse_2);
    }
    getPerimeter() {    // Периметр
        return this.a + this.b + this.c;
    }
    getArea() {         // Площадь
        let p = this.getPerimeter() / 2;
        return Math.sqrt(p * (p-this.a) * (p-this.b) * (p-this.c));
    }
    toString() {
        return `triangle: ${this.a}, ${this.b}, ${this.c}`;
    }
}
```

Тесты

Тест №1:

```
function test1() {
    let tr = new Triangle(0, 1, 1);
    console.log("Is " + tr.toString() + " exist - " + tr.isExist());

    tr = new Triangle(-1, 1, 1);
    console.log("Is " + tr.toString() + " exist - " + tr.isExist());

    tr = new Triangle(3, 1, 1);
    console.log("Is " + tr.toString() + " exist - " + tr.isExist());

    tr = new Triangle(2, 3, 1);
    console.log("Is " + tr.toString() + " exist - " + tr.isExist());

    tr = new Triangle(2, 3, 2);
    console.log("Is " + tr.toString() + " exist - " + tr.isExist());
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_2_2.js

Is triangle: 0, 1, 1 exist - false
Is triangle: -1, 1, 1 exist - false
Is triangle: 3, 1, 1 exist - false
Is triangle: 2, 3, 1 exist - false
Is triangle: 2, 3, 2 exist - true
```

Тест №2:

```
function test2() {
    let tr = new Triangle(2, 3, 1.01);
    console.log(tr.toString());
    console.log("Is right - " + tr.isRight());
    console.log("P = " + tr.getPerimeter());
    console.log("S = " + tr.getArea());
    console.log();

    tr = new Triangle(3, 5, 4);
    console.log(tr.toString());
    console.log("Is right - " + tr.isRight());
    console.log("P = " + tr.getPerimeter());
    console.log("S = " + tr.getArea());
    console.log();

    tr = new Triangle(1, 1, 1);
    console.log(tr.toString());
    console.log("Is right - " + tr.isRight());
    console.log("P = " + tr.getPerimeter());
    console.log("S = " + tr.getArea());
    console.log();

    tr = new Triangle(8, 6, 10);
    console.log(tr.toString());
    console.log("Is right - " + tr.isRight());
    console.log("P = " + tr.getPerimeter());
    console.log("S = " + tr.getArea());
    console.log();
}
```

Результат:

```
PS D:\Work\JS-bmstu\lab_1> node .\task_2_2.js

triangle: 2, 3, 1.01
Is right - false
P = 6.01
S = 0.1735648275861194

triangle: 3, 5, 4
Is right - true
P = 12
S = 6

triangle: 1, 1, 1
Is right - false
P = 3
S = 0.4330127018922193

triangle: 8, 6, 10
Is right - true
P = 24
S = 24
```

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.

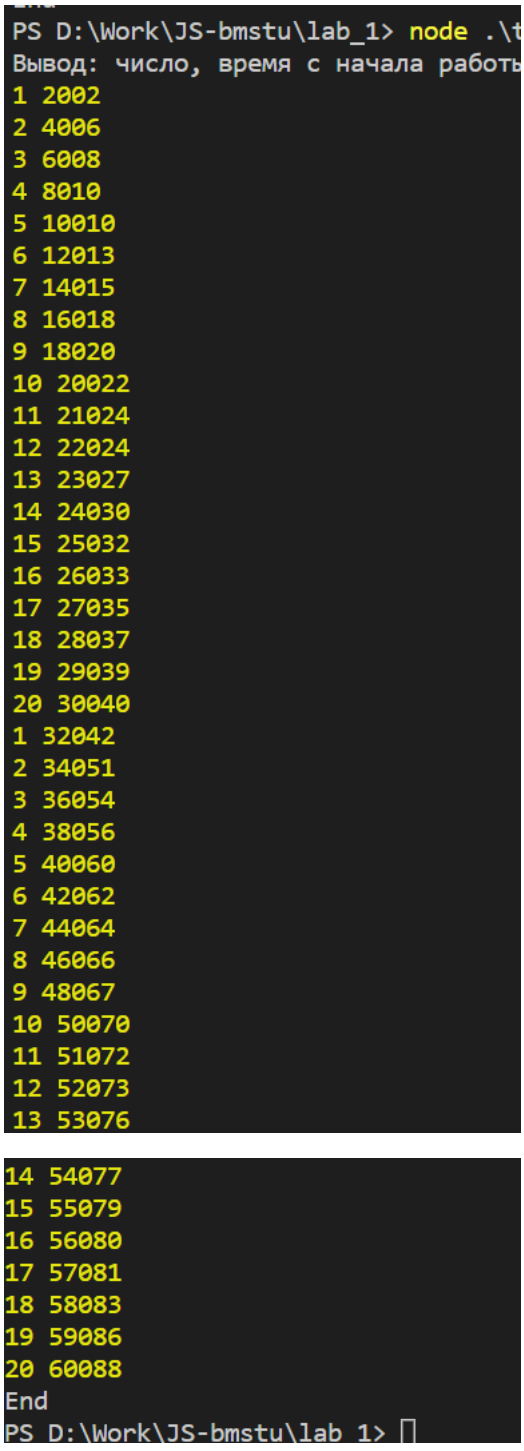
Это должно происходить циклически.

Программная реализация

```
"use strict";

let interval;
let begin_t;
let func_n = 0;
let n = 1;
let func_arr = [f1, f2, f1, f2, fstop];
let delay_arr = [2000, 1000, 2000, 1000, 0];

function f1() {
    console.log(n, Date.now() - begin_t);
    n++;
    if (n >= 11)
    {
        func_n++;

        clearInterval(interval);
        interval = setInterval(func_arr[func_n], delay_arr[func_n]);
    }
}
function f2() {
    console.log(n, Date.now() - begin_t);
    n++;

    if (n >= 21)
    {
        n = 1;
        func_n++;

        clearInterval(interval);
        interval = setInterval(func_arr[func_n], delay_arr[func_n]);
    }
}
function fstop() {
    console.log("End");
    clearInterval(interval);
}
function main() {
    console.log("Вывод: число, время с начала работы (мс)");
    begin_t = Date.now();
    interval = setInterval(func_arr[func_n], delay_arr[func_n]);
}

main();
```

Тесты

Результат (для отражения задержек во времени в отчёте, выводится время, прошедшее с запуска функции в мс):



Вывод

В рамках лабораторной работы было выполнено ознакомление и практическое закрепление основ языка JS.