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»

Task № _1_

Дисциплина: Архитектура ЭВМ

Тема : Знакомство с NodeJS. Функции.

Студент _Расколотов Д.Ю.

Группа _ИУ7-54Б_____

Преподаватель _Повов А.Ю.

Москва.
2020 г.

Цель работы: Познакомиться с NodeJS. Изучить темы: Объекты, Функции, Преобразование типов, Строки.

Задание 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

Код программы:

```
"use strict";
class Student {
  name;
  age;
  constructor(name, age) {
    this.age = age;
    this.name = name;
  }

  output() {
    console.log(this.name + ": " + this.age);
  }
}
```

```
class Students {
  arr;

  constructor() {
    this.arr = [];
  }

  addStudent(st) {
    for (const elem of this.arr) {
      if (elem.name === st.name) {
        return -1;
      }
    }
    this.arr.push(st);
    return 0;
  }

  output() {
    for (const elem of this.arr) {
      elem.output();
    }
  }
  upDate(st, nst, nage) {
    if (this.arr.length === 0) {
      return -1;
    }
    for (const elem of this.arr) {
      if (st === elem.name) {
        elem.name = nst;
        elem.age = nage;
      }
    }
  }

  Delete(st) {
    if (this.arr.length === 0) {
      return -1;
    }
    let i = 0;
    for (const elem of this.arr) {
```

```
      i++;
      if (st === elem.name) {
        this.arr.splice((i-1), 1);
      }
    }
  }

  getAverageAge() {
    if (this.arr.length === 0) {
      return -1;
    }
    let sum = 0;
    for (const elem of this.arr) {
      sum += elem.age;
    }
    return sum / this.arr.length;
  }

  getHighAge() {
    if (this.arr.length === 0) {
      return -1;
    }

    let maxAge = this.arr[0].age;
    let max = 0;

    for (const elem of this.arr) {
      if (elem.age > maxAge) {
        maxAge = elem.age;
        max = elem;
      }
    }
    console.log("Самый старый ребёнок-  " + max.name + " : " +
max.age);
  }

  getPeriodAge(minage, maxage) {
    if (this.arr.length === 0) {
      return -1;
    }
```

```
    for (const elem of this.arr) {
      if (elem.age > minage && elem.age < maxage) {
        console.log(
          "В диапозон значений от " +
            minage +
            " до " +
            maxage +
            " входит-" +
            elem.name +
            " : " +
            elem.age
        );
      }
    }
  }

  getFirstB(simb) {
    if (this.arr.length === 0) {
      return -1;
    }
    for (const elem of this.arr) {
      if (elem.name[0] === simb) {
        console.log(
          "Ребёнок начинающ. с буквы " +
            simb +
            " вот-" +
            elem.name +
            " : " +
            elem.age
        );
      }
    }
  }

  getLongLastN(ln) {
    if (this.arr.length === 0) {
      return -1;
    }
    for (const elem of this.arr) {
      if (elem.name.length > ln) {
```

```
        console.log(
          "Ребёнок с длинной фамилией " + elem.name + " : " + e
lem.age
        );
      }
    }
  }

  getAtoY() {
    if (this.arr.length === 0) {
      return -1;
    }
    const vowels = ["A", "E", "I", "O", "U"];
    for (const elem of this.arr) {
      if (vowels.includes(elem.name[0])) {
        console.log(
          "Ребёнок с гласной в начале " + elem.name + " : " + e
lem.age
        );
      }
    }
  }
  getDeleteSt() {
    if (this.arr.length === 0) {
      return -1;
    }
    this.arr.pop();
    return 0;
  }
}

function testSession() {
  const student = new Student("Seryogina", 4);
  const student2 = new Student("Kozachenko", 15);
  const student3 = new Student("Ivanov", 17);
  const student4 = new Student("Opolov", 19);
  const student5 = new Student("Opol", 40);
  const StudentsArr = new Students();
  StudentsArr.addStudent(student);
  StudentsArr.addStudent(student2);
```

```
  StudentsArr.addStudent(student3);
  StudentsArr.addStudent(student4);
  StudentsArr.addStudent(student5);
  StudentsArr.output();
  console.log(StudentsArr.arr.length);
  console.log(StudentsArr.getAverageAge());
  StudentsArr.getHighAge();
  StudentsArr.upDate("Ivanov", "Poluna", 15);
  StudentsArr.output();
  StudentsArr.Delete("Opolov");
  StudentsArr.getPeriodAge(5, 19);
  StudentsArr.getFirstB("P");
  StudentsArr.getLongLastN(6);
  StudentsArr.getAtoY();
  StudentsArr.output();
}

testSession();
```

Тестовая часть:

1.

```
const student = new Student("Seryogina", 4);
  const student2 = new Student("Kozachenko", 15);
  const student3 = new Student("Ivanov", 17);
  const student4 = new Student("Opolov", 19);
  const student5 = new Student("Opol", 40);
  const StudentsArr = new Students();
  StudentsArr.addStudent(student);
  StudentsArr.addStudent(student2);
  StudentsArr.addStudent(student3);
  StudentsArr.addStudent(student4);
  StudentsArr.addStudent(student5);
```

Вывод:

```
Seryogina: 4
Kozachenko: 15
Ivanov: 17
Opolov: 19
Opol: 40
```

2.

```
console.log(StudentsArr.getAverageAge());
StudentsArr.getHighAge();
StudentsArr.upDate("Ivanov", "Poluna", 15);
StudentsArr.output();
```

Вывод:

```
19
Самый старый ребёнок-  Opol : 40
Seryogina: 4
Kozachenko: 15
Poluna: 15
Opolov: 19
Opol: 40
```

3.

```
StudentsArr.getPeriodAge(5, 19);
StudentsArr.getFirstB("P");
StudentsArr.getLongLastN(6);
StudentsArr.getAtoY();
StudentsArr.Delete("Opolov");
StudentsArr.output();
```

Вывод:

```
В диапозон значений от 5 до 19 входит-Kozachenko : 15
В диапозон значений от 5 до 19 входит-Poluna : 15
Ребёнок начинающ. с буквы P вот-Poluna : 15
Ребёнок с длинной фамилией Seryogina : 4
Ребёнок с длинной фамилией Kozachenko : 15
Ребёнок с гласной в начале Opolov : 19
Ребёнок с гласной в начале Opol : 40
Seryogina: 4
Kozachenko: 15
Poluna: 15
Opol: 40
```

Задание 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

Код программы:

```
class Student {
  constructor(gr_name, st_card, marks) {
    this.gr_name = gr_name;
    this.st_card = st_card;
    this.marks = marks;
  }

  get output() {
    return `${this.gr_name} ${this.st_card} ${this.marks}`;
  }
}

class Students {
  constructor() {
    this.arr = [];
  }

  CREATE(st) {
    for (const man of this.arr) {
      if (man.st_card === st.st_card) {
```

```
      return false;
    }
  }
  this.arr.push(st);
  return true;
}

READ(name) {
  for (const man of this.arr) {
    if (name === man.st_card) {
      console.log(man.output);
    }
  }
  return true;
}

UPDATE(st) {
  let j = -1;
  for (let i = 0; i < this.arr.length; i++) {
    if (this.arr[i].st_card === st.st_card) {
      j = i;
    }
  }
  if (j !== -1) {
    console.log("Invalid read student card");
    return false;
  }
  this.arr.splice(j, 1, st);
  return true;
}

DELETE(st) {
  let j = -1;
  for (let i = 0; i < this.arr.length; i++) {
    if (this.arr[i].gr_name === st.gr_name) {
      j = i;
    }
  }
  if (j !== -1) {
    console.log("Invalid read student card");
```

```
    return false;
  }
  this.arr.splice(j, 1);
  return true;
}

average_mark_st(st) {
  let n = 0;
  let i = 0;
  for (const mark of st.marks) {
    i++;
    n += mark;
  }
  if (i === 0) {
    console.log("Invalid read marks");
    return false;
  }
  return n / i;
}

info_st_group(group) {
  let flag = 0;
  for (const man of this.arr) {
    if (man.gr_name === group) {
      flag = 1;
      console.log(man.output);
    }
  }
  if (flag === 0) {
    console.log("Invalid read info student group");
    return false;
  }
  return true;
}

info_best_st_group(gr_name) {
  let len = -1;
  let best_st = undefined;
  for (const man of this.arr) {
    if (man.marks.length > len && man.gr_name === gr_name) {
```

```
        len = man.marks.length;
        best_st = man;
      }
    }
    if (len === -1) {
      console.log("Invalid read info student group");
      return false;
    }
    console.log(best_st.output);
    return true;
  }

  info_zero_st() {
    let flag = 0;
    for (const man of this.arr) {
      if (man.marks.length === 0) {
        flag = 1;
        console.log(man.output);
      }
    }

    if (flag !== 0) {
      console.log("Invalid read student card");
      return false;
    }

    return true;
  }
}

function main() {
  let st1 = new Student("abc", 123, [4, 3, 2]);
  let st11 = new Student("abc", 1323, [1, 3, 2]);
  let st2 = new Student("qwerty", 523123133, [5, 5, 2]);
  let st3 = new Student("rtyuifgh", 565, [4, 2, 2]);
  let st4 = new Student("rqwe", 55, [4, 2, 2]);

  let sts = new Students();

  sts.CREATE(st1);
```

```
  sts.CREATE(st11);
  sts.CREATE(st2);
  sts.CREATE(st3);

  sts.DELETE(st4);

  console.log(sts.READ(st1));

  let st5 = new Student("qwerty", 5233, [5, 5, 5]);
  sts.UPDATE(st5);
  console.log(sts.READ(st2));

  console.log("Среднее значение: " + sts.average_mark_st(st1));

  console.log("Информация: " + sts.info_st_group("abc"));

  console.log("The best one: " + sts.info_best_st_group("abc"))
;

  let st6 = new Student("rqwe", 55, []);
  sts.CREATE(st6);
  console.log(sts.info_zero_st());
}

main();
```

Тестовая часть:

```
console.log("Среднее значение: " + sts.average_mark_st(st1));
console.log("Информация: " + sts.info_st_group("abc"));
console.log("The best one: " + sts.info_best_st_group("abc"));
let st6 = new Student("rqwe", 55, []);
sts.CREATE(st6);
console.log(sts.info_zero_st());
```

Вывод:

```
Среднее значение: 3
abc 123 4,3,2
abc 1323 1,3,2
Информация: true
abc 123 4,3,2
The best one: true
rqwe 55
```

Задание 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

Код программы:

```
"use strict"

class Dot {
    constructor(name, x, y) {
        this.name = name
        this.x = x
        this.y = y
    }

    get output(){
        return `${this.name} ${this.x} ${this.y}`;
    }
}

class Dots{
    constructor() {
        this.arr = []
    }

    CREATE(dt){
        for (const d of this.arr){
```

```
            if (dt.name === d.name){
                return false
            }
        }
        this.arr.push(dt)
        return true
    }

    READ(dt){
        console.log(dt.output)
        return true
    }

    UPDATE(dt){
        let j = -1
        for (let i = 0; i < this.arr.length; i++){
            if (this.arr[i].name === dt.name){
                j = i
            }
        }
        if (j === -1) {
            console.log("Invalid read dot name")
            return false
        }
        this.arr.splice(j, 1, dt)
        return true
    }

    DELETE(dt){
        let j = -1
        for (let i = 0; i < this.arr.length; i++){
            if (this.arr[i].name === dt.name){
                j = i
            }
        }
        if (j === -1) {
            console.log("Invalid read dot name")
            return false
        }
        this.arr.splice(j, 1)
```

```
        return true
    }

    max_range(){
        if (this.arr.length <= 1){
            console.log("Invalid read array")
            return false
        }
        let max = 0
        let temp_max = 0
        let log = []
        for (let i = 0; i < this.arr.length - 1; i++){
            temp_max = Math.sqrt(Math.pow(this.arr[i].x - this.
arr[i + 1].x, 2) +
                Math.pow(this.arr[i].y - this.arr[i + 1].y, 2))
            if (temp_max > max){
                max = temp_max
                log[0] = this.arr[i]
                log[1] = this.arr[i + 1]
            }
        }
        return log
    }

    near_dots(dt, len){
        if (len <= 0) {
            console.log("Invalid read len")
            return false
        }
        let temp_len = 0
        let log = []
        for (let i = 0; i < this.arr.length; i++){
            temp_len = Math.sqrt(Math.pow(dt.x - this.arr[i].x,
 2) + Math.pow(dt.y - this.arr[i].y, 2))
            if (temp_len < len){
                log.push(this.arr[i])
            }
        }
        return log
    }
```

```
    get_dot_axes(d){
        let log = []
        for (let i = 0; i < this.arr.length; i++) {
            if (this.arr[i].x > d.x && this.arr[i].y > d.y){
                log.push(this.arr[i], "1")
            }
            else if (this.arr[i].x < d.x && this.arr[i].y > d.y
){
                log.push(this.arr[i], "2")
            }
            else if (this.arr[i].x < d.x && this.arr[i].y < d.y
){
                log.push(this.arr[i], "3")
            }
            else if (this.arr[i].x > d.x && this.arr[i].y < d.y
){
                log.push(this.arr[i], "4")
            }
        }
        return log
    }

    inside_rectangle(dt_l_d, dt_r_u){
        let log = []
        for (let i = 0; i < this.arr.length; i++) {
            if (this.arr[i].x < dt_r_u.x && this.arr[i].x > dt_
l_d.x &&
                this.arr[i].y < dt_r_u.y && this.arr[i].y > dt_
l_d.y)
                log.push(this.arr[i])
            }
        return log
    }

}

function main(){
    let d1 = new Dot("abc", 123, 1)
    let d11 = new Dot("ab12c", 1323, 3213)
```

```
    let d2 = new Dot("qwerty", 523123133, 1)
    let d3 = new Dot("rtyuifgh", 565, 334)
    let d4 = new Dot("rqwe", 55, 23)

    let ds = new Dots()

    ds.CREATE(d1)
    ds.CREATE(d11)
    ds.CREATE(d2)
    ds.CREATE(d3)
    ds.CREATE(d4)

    ds.DELETE(d4)

    console.log(ds.READ(d1))

    let d5 = new Dot("qwerty", 5233, 12)
    ds.UPDATE(d5)

    console.log(ds.READ(d2))

    console.log(ds.max_range())

    console.log(ds.near_dots(d1, 1000))

    console.log(ds.get_dot_axes(d2))

    console.log(ds.inside_rectangle(d1, d11))
}

main()
```

Тестовая часть:

```
console.log(ds.max_range())
console.log(ds.near_dots(d1, 1000))
console.log(ds.get_dot_axes(d2))
console.log(ds.inside_rectangle(d1, d11))
```

Вывод:

```
[
  Dot { name: 'ab12c', x: 1323, y: 3213 },
  Dot { name: 'qwerty', x: 5233, y: 12 }
]
[
  Dot { name: 'abc', x: 123, y: 1 },
  Dot { name: 'rtyuifgh', x: 565, y: 334 }
]
[
  Dot { name: 'ab12c', x: 1323, y: 3213 },
  '2',
  Dot { name: 'qwerty', x: 5233, y: 12 },
  '2',
  Dot { name: 'rtyuifgh', x: 565, y: 334 },
  '2'
]
[ Dot { name: 'rtyuifgh', x: 565, y: 334 } ]
```

Заключительная часть:

Научился работать с объектами, функциями и строками в NodeJS, попрактиковался на приведенных выше задачах.