|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки 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 \_\_\_\_\_\_

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

Дисциплина Архитектура ЭВМ

Студент Куликов Дмитрий Алексеевич

*фамилия, имя, отчество*

Группа ИУ7-52Б

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Попов А.Ю.

*подпись, дата фамилия, и.о.*

*2020 г.*

**Цель работы:**

Приобретение и закрепление на практике следующих навыков:

1. Взаимодействие между серверами. Сервер для отправки запросов на другой сервер.

2. Передача параметров скрипту.

3. Дочерние процессы.

4. Знакомство с Prolog.

# Task 7

# Задание 1

**Условие задачи:**

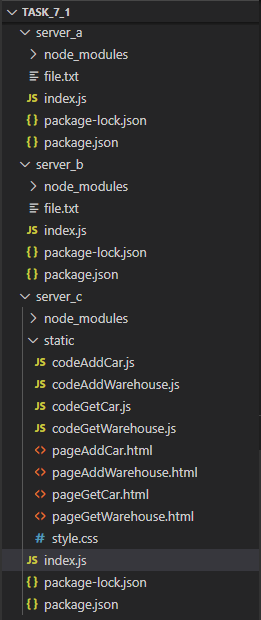
Создать сервер **А**. На стороне сервера хранится файл с содержимым в формате **JSON**. При получении запроса на **/insert/record** идёт добавление записи в файл. При получении запроса на **/select/record** идёт получение записи из файла. Каждая запись хранит информацию о машине (*название* и *стоимость*).

Создать сервер **Б**. На стороне сервера хранится файл с содержимым в формате **JSON**. Каждая запись в файле хранит информацию о складе и массиве машин, находящихся на данном складе. То есть каждая запись хранит в себе название склада (*строку*) и массив названий машин (*массив строк*). При получении запроса на **/insert/record** идёт добавление записи в файл. При получении запроса на **/select/record** идёт получение записи из файла.

Создать сервер **C**. Сервер выдаёт пользователю страницы с формами для ввода информации. При этом сервер взаимодействует с серверами **А** и **Б**. Реализовать для пользователя функции:

* создание нового типа машины
* получение информации о стоимости машины по её типу
* создание нового склада с находящимися в нём машинами
* получение информации о машинах на складе по названию склада

Реализовать удобный для пользователя интерфейс взаимодействия с системой (использовать поля ввода и кнопки).



**Листинг:**

**server\_a**

index.js

"use strict";

// импорт библиотеки

const express = require("express");

const fs = require("fs");

// запускаем сервер

const app = express();

const port = 5002;

app.listen(port);

console.log("Server on port " + port);

// заголовки для ответа

app.use(function(req, res, next) {

    res.header("Cache-Control", "no-cache, no-store, must-revalidate");

    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    res.header("Access-Control-Allow-Origin", "\*");

    next();

});

// загрузка тела

function loadBody(request, callback) {

    let body = [];

    request.on('data', (chunk) => {

        body.push(chunk);

    }).on('end', () => {

        body = Buffer.concat(body).toString();

        callback(body);

    });

}

// приём запроса

app.post("/insert/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const car = obj.car;

        const price = obj.price;

        let contentString = fs.readFileSync("./file.txt", "utf8");

        let obj1 = [];

        if (contentString != '') {

            obj1 = JSON.parse(contentString);

        }

        obj1.push({"car": car, "price": price})

        contentString = JSON.stringify(obj1);

        fs.writeFileSync("./file.txt", contentString);

        response.end(JSON.stringify({

            answer: "car was added"

        }));

    });

});

// приём запроса

app.post("/select/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const price = obj.price;

        let contentString = fs.readFileSync("./file.txt", "utf8");

        let answerString = 'Not this car';

        if (contentString != '') {

            let obj1 = JSON.parse(contentString);

            for (let i = 0; i < obj1.length; i++) {

                if (obj1[i].price == price) {

                    answerString = `car: ${obj1[i].car} price: ${obj1[i].price}`;

                    break;

                }

            }

        }

        response.end(JSON.stringify({

            answer: answerString

        }));

    });

});

**server\_b**

index.js

"use strict";

// импорт библиотеки

const express = require("express");

const fs = require("fs");

// запускаем сервер

const app = express();

const port = 5000;

app.listen(port);

console.log("Server on port " + port);

// заголовки для ответа

app.use(function(req, res, next) {

    res.header("Cache-Control", "no-cache, no-store, must-revalidate");

    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    res.header("Access-Control-Allow-Origin", "\*");

    next();

});

// загрузка тела

function loadBody(request, callback) {

    let body = [];

    request.on('data', (chunk) => {

        body.push(chunk);

    }).on('end', () => {

        body = Buffer.concat(body).toString();

        callback(body);

    });

}

// приём запроса

app.post("/insert/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const warehouse = obj.warehouse;

        let arrCar = obj.arrCar;

        let contentString = fs.readFileSync("./file.txt", "utf8");

        let obj1 = [];

        if (contentString != '') {

            obj1 = JSON.parse(contentString);

        }

        arrCar = arrCar.split(",");

        obj1.push({"warehouse": warehouse, "arrCar": arrCar})

        contentString = JSON.stringify(obj1);

        fs.writeFileSync("./file.txt", contentString);

        response.end(JSON.stringify({

            answer: "warehouse was added"

        }));

    });

});

// приём запроса

app.post("/select/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const warehouse = obj.warehouse;

        let contentString = fs.readFileSync("./file.txt", "utf8");

        let answerString = 'Not this warehouse';

        if (contentString != '') {

            let obj1 = JSON.parse(contentString);

            for (let i = 0; i < obj1.length; i++) {

                if (obj1[i].warehouse == warehouse) {

                    answerString = `cars:`;

                    for (let j = 0; j < obj1[i].arrCar.length; j++) {

                        if (j != obj1[i].arrCar.length - 1) {

                            answerString += `${obj1[i].arrCar[j]},`;

                        } else {

                            answerString += `${obj1[i].arrCar[j]}`;

                        }

                    }

                    break;

                }

            }

        }

        response.end(JSON.stringify({

            answer: answerString

        }));

    });

});

**server\_c**

index.js

"use strict";

// импорт библиотек

const express = require("express");

const request = require("request");

// запускаем сервер

const app = express();

const port = 5004;

app.listen(port);

console.log(`Server on port ${port}`);

const way = \_\_dirname + "/static";

app.use(express.static(way));

// заголовки в ответ клиенту

app.use(function(req, res, next) {

    res.header("Cache-Control", "no-cache, no-store, must-revalidate");

    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    res.header("Access-Control-Allow-Origin", "\*");

    next();

});

// функция для отправки POST запроса на другой сервер

function sendPost(url, body, callback) {

    // задаём заголовки

    const headers = {};

    headers["Cache-Control"] = "no-cache, no-store, must-revalidate";

    headers["Connection"] = "close";

    // отправляем запрос

    request.post({

        url: url,

        body: body,

        headers: headers,

    }, function (error, response, body) {

        if(error) {

            callback(null);

        } else {

            callback(body);

        }

    });

}

// принимаем GET запрос и отправляем POST запрос на другой сервер

app.get("/addCar", function(request, response) {

    const c = request.query.car;

    const p = request.query.price;

    sendPost("http://localhost:5002/insert/record", JSON.stringify({

        car: c,

        price: p

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const answer = answerObject.answer;

        response.end("Answer: " + answer);

    });

});

// принимаем GET запрос и отправляем POST запрос на другой сервер

app.get("/getCar", function(request, response) {

    const p = request.query.price;

    sendPost("http://localhost:5002/select/record", JSON.stringify({

        price: p

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const answer = answerObject.answer;

        response.end("Answer: " + answer);

    });

});

// принимаем GET запрос и отправляем POST запрос на другой сервер

app.get("/addWarehouse", function(request, response) {

    const w = request.query.warehouse;

    const ac = request.query.arrCar;

    sendPost("http://localhost:5000/insert/record", JSON.stringify({

        warehouse: w,

        arrCar: ac

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const answer = answerObject.answer;

        response.end("Answer: " + answer);

    });

});

// принимаем GET запрос и отправляем POST запрос на другой сервер

app.get("/getWarehouse", function(request, response) {

    const w = request.query.warehouse;

    sendPost("http://localhost:5000/select/record", JSON.stringify({

        warehouse: w

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const answer = answerObject.answer;

        response.end("Answer: " + answer);

    });

});

pageAddCar.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Добавить машину</title>

    <link rel="stylesheet" href="/style.css">

</head>

<body>

    <p>Машина</p>

    <input id="field-car" type="text" spellcheck="false" autocomplete="off">

    <p>Стоимость</p>

    <input id="field-price" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <div id="btn-send" class="btn-class">Отправить</div>

    <br>

    <br>

    <script src="/codeAddCar.js"></script>

</body>

</html>

codeAddCar.js

"use strict";

window.onload = function() {

    // input fields

    const car = document.getElementById("field-car");

    const price = document.getElementById("field-price");

    const btn = document.getElementById("btn-send");

    // ajax get

    function ajaxGet(urlString, callback) {

        let r = new XMLHttpRequest();

        r.open("GET", urlString, true);

        r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");

        r.send(null);

        r.onload = function() {

            callback(r.response);

        };

    };

    // click event

    btn.onclick = function() {

        const c = car.value;

        const p = price.value;

        const url = `/addCar?car=${c}&price=${p}`;

        ajaxGet(url, function(stringAnswer) {

            alert(stringAnswer);

        });

    };

}

pageGetCar.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Получить машину</title>

    <link rel="stylesheet" href="/style.css">

</head>

<body>

    <p>Стоимость</p>

    <input id="field-price" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <div id="btn-send" class="btn-class">Отправить</div>

    <br>

    <br>

    <h1 id="result-label"></h1>

    <script src="/codeGetCar.js"></script>

</body>

</html>

codeGetCar.js

"use strict";

window.onload = function() {

    // input fields

    const price = document.getElementById("field-price");

    const btn = document.getElementById("btn-send");

    // label

    const label = document.getElementById("result-label");

    // ajax get

    function ajaxGet(urlString, callback) {

        let r = new XMLHttpRequest();

        r.open("GET", urlString, true);

        r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");

        r.send(null);

        r.onload = function() {

            callback(r.response);

        };

    };

    // click event

    btn.onclick = function() {

        const p = price.value;

        const url = `/getCar?price=${p}`;

        ajaxGet(url, function(stringAnswer) {

            label.innerHTML = `${stringAnswer}`;

        });

    };

}

pageAddWarehouse.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Добавить склад машин</title>

    <link rel="stylesheet" href="/style.css">

</head>

<body>

    <p>Склад</p>

    <input id="field-warehouse" type="text" spellcheck="false" autocomplete="off">

    <p>Массив машин, перечислить через запятую без пробелов</p>

    <input id="field-array-car" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <div id="btn-send" class="btn-class">Отправить</div>

    <br>

    <br>

    <script src="/codeAddWarehouse.js"></script>

</body>

</html>

codeAddWarehouse.js

"use strict";

window.onload = function() {

    // input fields

    const warehouse = document.getElementById("field-warehouse");

    const arrCar = document.getElementById("field-array-car");

    const btn = document.getElementById("btn-send");

    // ajax get

    function ajaxGet(urlString, callback) {

        let r = new XMLHttpRequest();

        r.open("GET", urlString, true);

        r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");

        r.send(null);

        r.onload = function() {

            callback(r.response);

        };

    };

    // click event

    btn.onclick = function() {

        const w = warehouse.value;

        const ac = arrCar.value;

        const url = `/addWarehouse?warehouse=${w}&arrCar=${ac}`;

        ajaxGet(url, function(stringAnswer) {

            alert(stringAnswer);

        });

    };

}

pageGetWarehouse.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Получить машины со склада</title>

    <link rel="stylesheet" href="/style.css">

</head>

<body>

    <p>Склад</p>

    <input id="field-warehouse" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <div id="btn-send" class="btn-class">Отправить</div>

    <br>

    <br>

    <h1 id="result-label"></h1>

    <script src="/codeGetWarehouse.js"></script>

</body>

</html>

codeGetWarehouse.js

"use strict";

window.onload = function() {

    // input fields

    const warehouse = document.getElementById("field-warehouse");

    const btn = document.getElementById("btn-send");

    // label

    const label = document.getElementById("result-label");

    // ajax get

    function ajaxGet(urlString, callback) {

        let r = new XMLHttpRequest();

        r.open("GET", urlString, true);

        r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");

        r.send(null);

        r.onload = function() {

            callback(r.response);

        };

    };

    // click event

    btn.onclick = function() {

        const w = warehouse.value;

        const url = `/getWarehouse?warehouse=${w}`;

        ajaxGet(url, function(stringAnswer) {

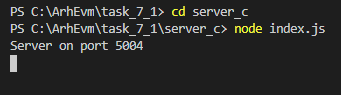
            label.innerHTML = `${stringAnswer}`;

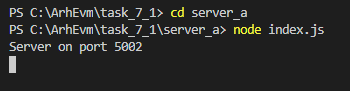
        });

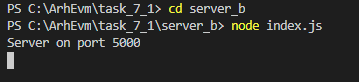
    };

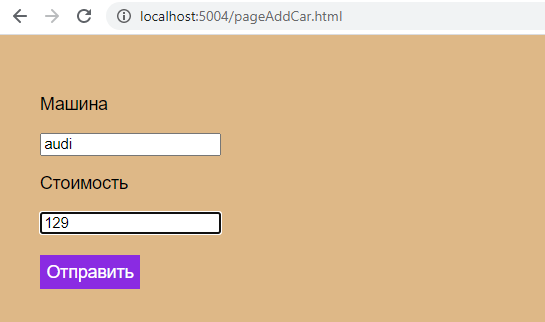
}

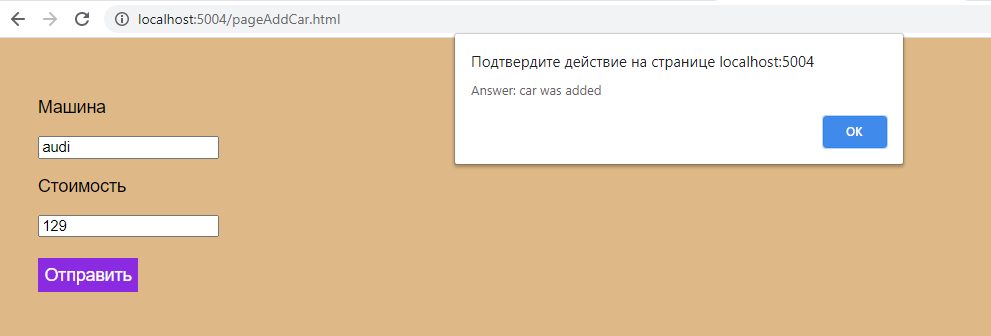
**Тестирование:**

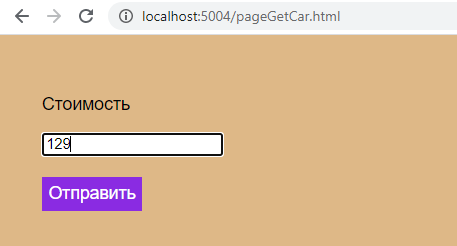
****

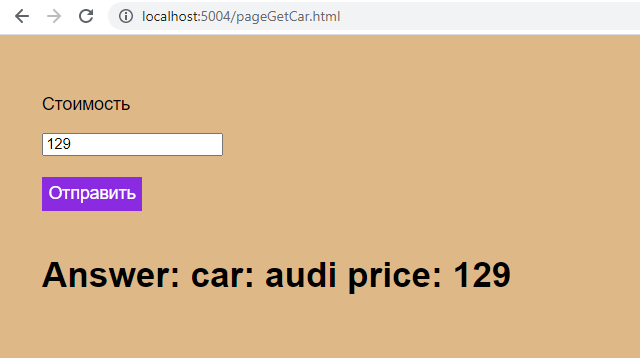
****

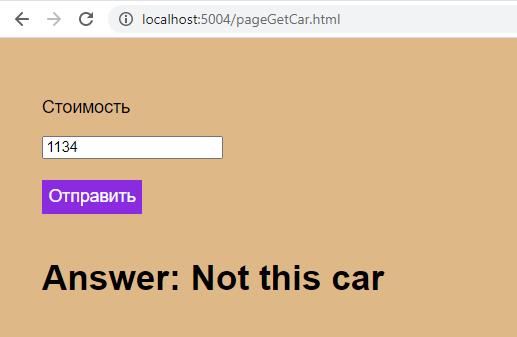
****

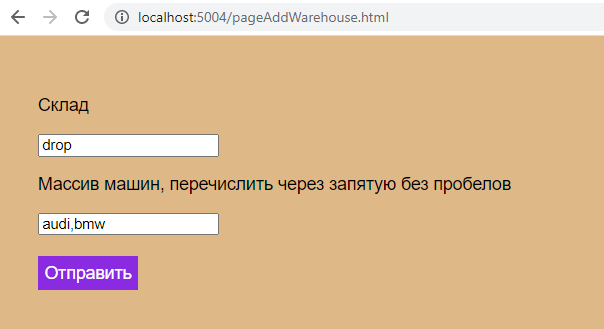
****

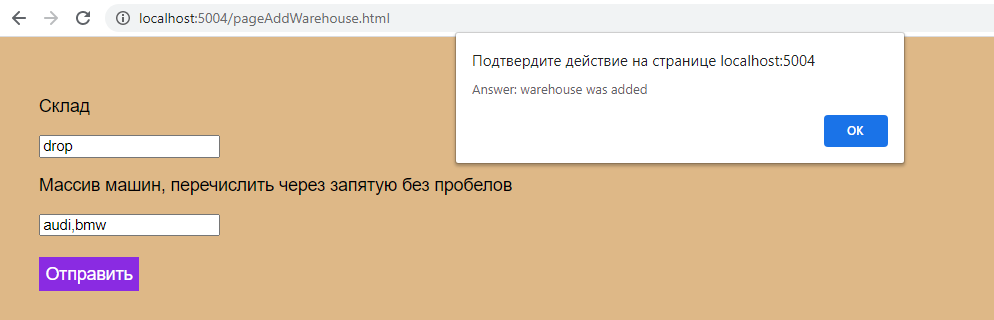
****

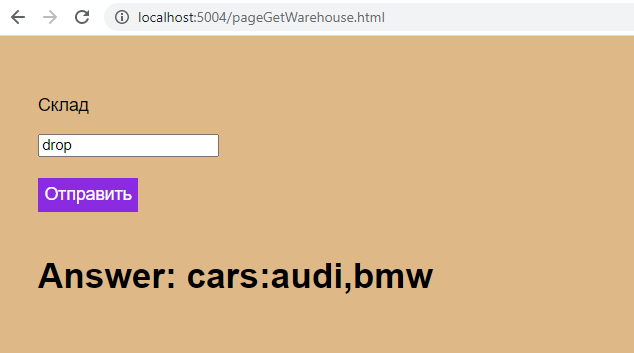
****

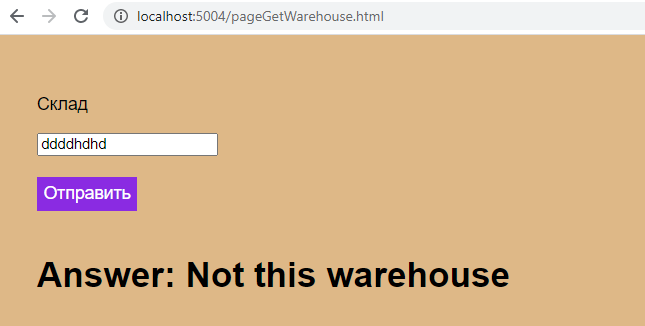
****

****

****

****

****

****

# 2. Задание 2

**Условие задачи:**

Написать скрипт, который принимает на вход число и считает его факториал. Скрипт должен получать параметр через **process.argv**.

Написать скрипт, который принимает на вход массив чисел и выводит на экран факториал каждого числа из массива. Скрипт принимает параметры через **process.argv**.

При решении задачи вызывать скрипт вычисления факториала через **execSync**.

**Листинг:**

index.js

"use strict";

const execSync = require('child\_process').execSync;

// функция для вызова программы и получения результата её работы

function useCmd(s) {

    const options = {encoding: 'utf8'};

    const cmd = s.toString();

    const answer = execSync(cmd, options);

    return answer.toString();

}

// получаем параметры скрипта

const type = "" + process.argv[2];

if (type === "number") {

    const number = "" + process.argv[3];

    // получаем факториал числа

    const factorialCommand = `node a.js ${number}`;

    console.log(factorialCommand);

    let factorial = useCmd(factorialCommand);

    factorial = parseInt(factorial);

    console.log(factorial);

} else if (type === "array") {

    let count = "" + process.argv[3];

    count = parseInt(count);

    let element;

    let array = [];

    for (let i = 4; i < count + 4 ; i++) {

        element = "" + process.argv[i];

        array.push(element);

    }

    // получаем факториал числа

    const factorialCommand = `node b.js ${array}`;

    console.log(factorialCommand);

    let factorial = useCmd(factorialCommand);

    console.log(factorial);

} else {

    console.log("incorrect key");

}

a.js

"use strict";

function factorial(n) {

    return n ? n \* factorial(n - 1) : 1;

}

const number = "" + process.argv[2];

const result = factorial(parseInt(number));

console.log("" + result);

b.js

"use strict";

function factorial(n) {

    return n ? n \* factorial(n - 1) : 1;

}

let number = process.argv[2];

number = number.split(",");

let result = "";

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

    number[i] = factorial(parseInt(number[i]));

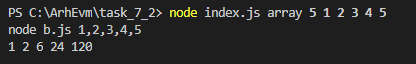
    result += number[i] + " ";

}

console.log(result);

**Тестирование:**

# 



# Task 8

# Задание 1

**Условие задачи:**

С клавиатуры считываются числа **A** и **B**. Необходимо вывести на экран все **числа Фибоначчи**, которые принадлежат отрезку от **A** до **B**.

**Листинг:**

ok.

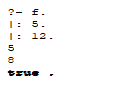
input(A, B) :- read(A), read(B); ok.

printA(A, S) :- A >= S, write(A), nl; ok.

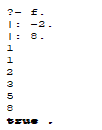
cicle(A, B, S, F) :- C is (A + B), printA(A, S), B =< F, cicle(B, C, S, F); ok.

f :- input(A, B), cicle(1, 1, A, B); ok.

**Тестирование:**

****

****

****

**Вывод:**

Все поставленные задачи были выполнены. Цель лабораторной работы достигнута.